The Resource Journal For IBM PC Programmers

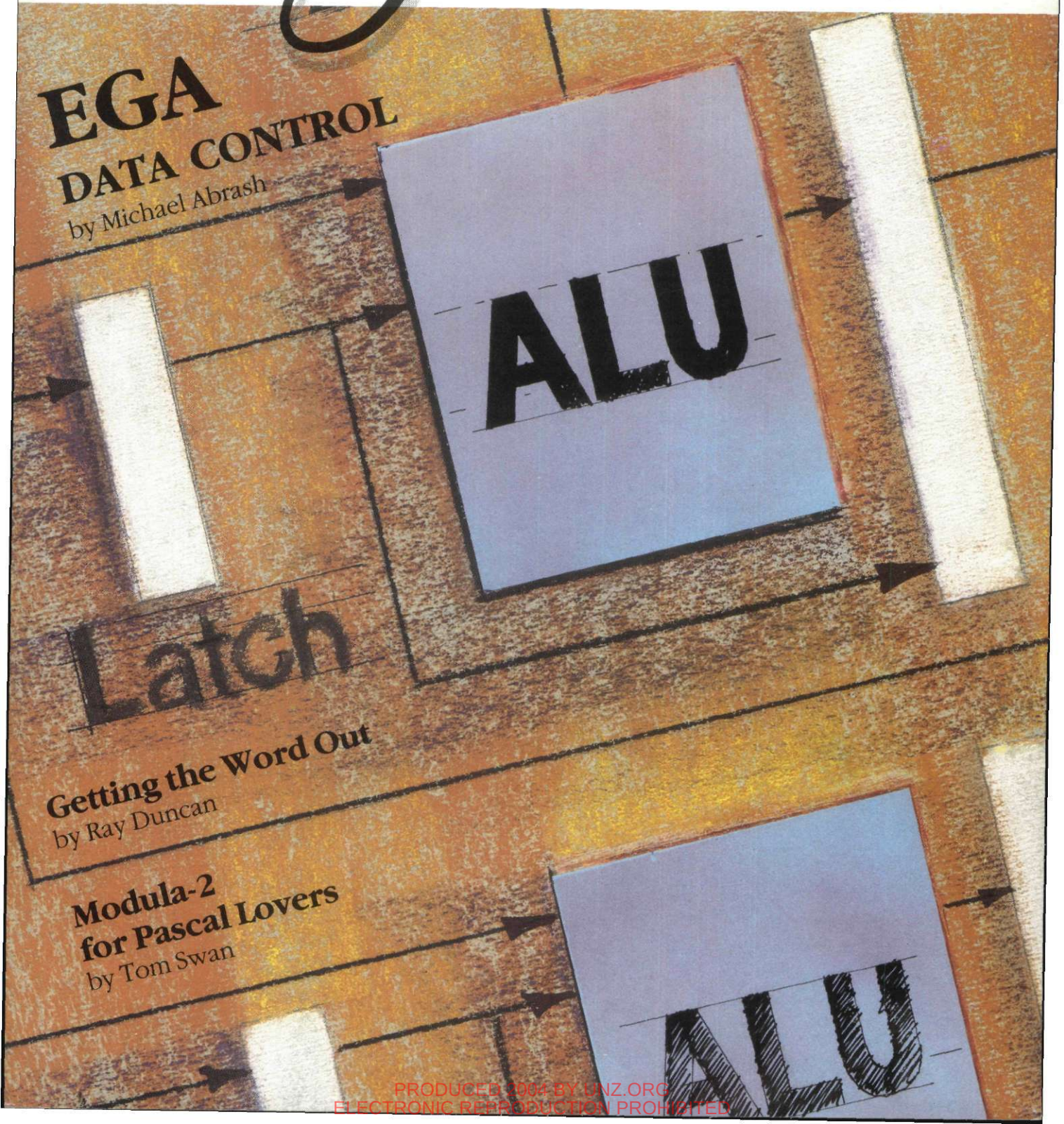
PROGRAMMER'S Journal

MAY/JUNE 1987
VOLUME 5.3
$3.95

EGA DATA CONTROL
by Michael Abrash

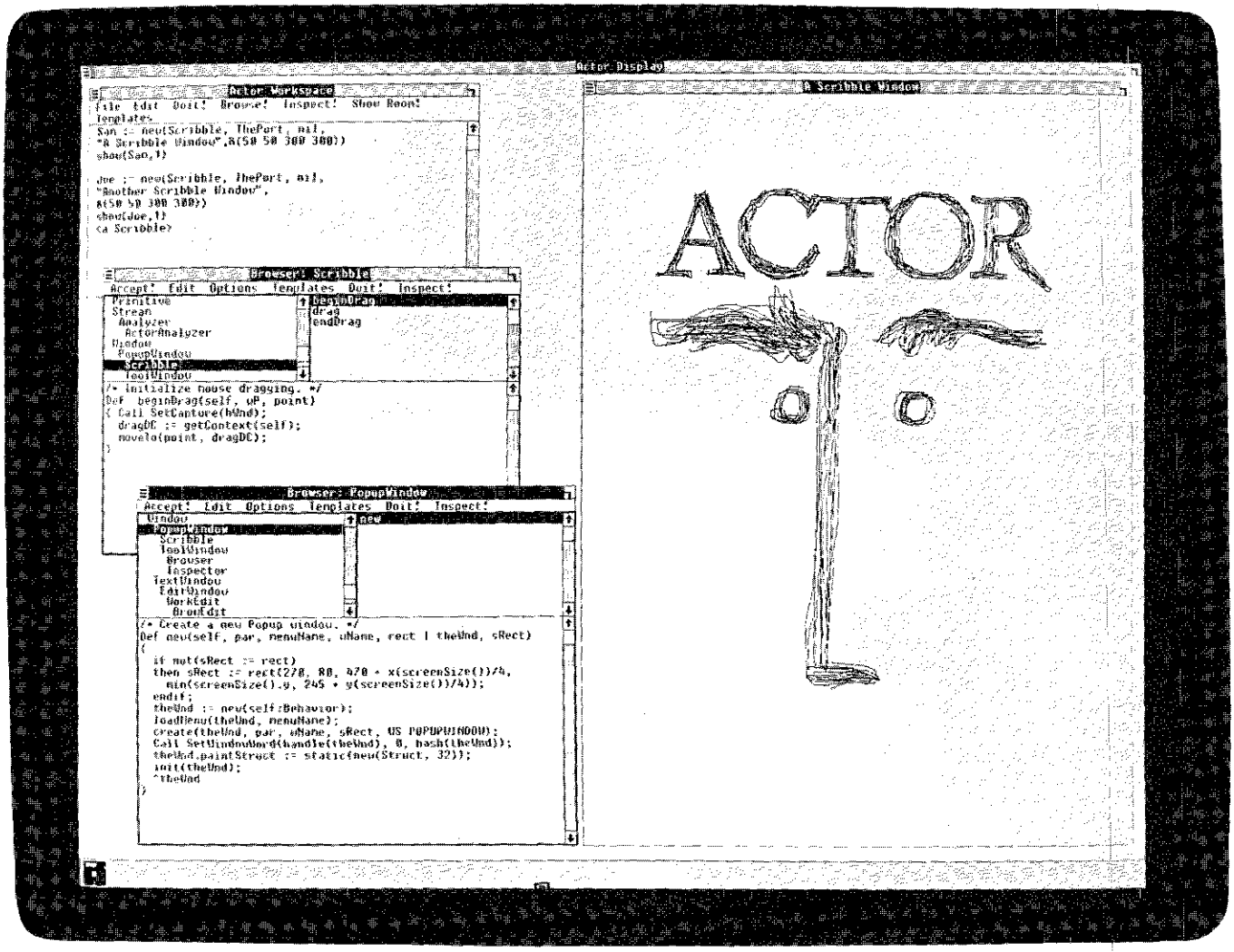
Getting the Word Out
by Ray Duncan

Modula-2 for Pascal Lovers
by Tom Swan



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

HOW TO WRITE A WINDOWS APPLICATION IN TEN MINUTES.



Actor™ is a new language that combines Microsoft® Windows with object-oriented programming. This means you can produce mouse and window applications very quickly.

For example, we created a simple "paint" program, and used it to draw the Actor logo you see on the screen. The whole program only took ten lines and ten minutes. Part of it is in the middle window on the left.

Above, you see the commands that initialized the paint window and made it appear on the screen. Below, some code that's built into Actor, specifying window behavior. Through a process known as "inheritance," it's called into play automatically.

Try programming in this new way, and you'll never go back.

Find out about Actor. Call The Whitewater Group®, (312) 491-2370.

Technology Innovation Center
906 University Place, Evanston, IL 60201

MICROSOFT WINDOWS

Circle Number 73 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

EVERYTHING THE C PROGRAMMER NEEDS

Eco-C88 C Compiler:

A full, professional C compiler with many ANSI enhancements at an unbelievably low price.

* Prototyping, enum, void data types, plus structure passing & assignment * All operators and data types * Over 200 library functions * cc and mini-make for easy use * 8087 support * ASM or OBJ output * Lint-like tiered error messages * Fast code * CED editor (edit-compile-link from within the editor) * Expanded user's manual * Not copy protected * All for only $59.95!

Eco-C88 Flexi-Graph Graphics Package

Everything you need to write dramatic graphics effects into your Eco-C88 C programs.

* EGA, CGA, and Z100 support * Over 100 graphics functions (many are PLOT-10 compatible) * Most assembler support routines are outside small model code-data * Write thru BIOS (for compatibility) or to memory (for speed) * Graphics function help from CED editor * World, pixel or turtle color graphics modes * 47 standard fill patterns, 17 line dashing patterns, Hershey fonts, plus user defineable fill, dash, and fonts * Supports view areas, rotateable fonts, clipping, arbitrary fill areas, extensive error checking, examples, and user's manual * Only $39.95

Eco-C88 Windowing Library

Use this library to build pop-up windows, help windows, selection menus, special effects—anywhere you need an attention getter.

* CGA and EGA support * Control any program that goes through the BIOS * Use up to 255 windows * No special window commands — use plain old printf( ) to write to a window * Resize and move windows * Custom window titles and borders * Can be used with ANSI device driver * Most window code-data are outside small model * User's manual and examples for only $29.95

Ecosoft Librarian

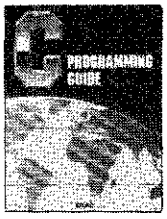
Combine your modules, functions, and subroutines into your own library for easy link commands. Compatible with any standard MSDOS OBJ files * Add, delete, and extract from a library * Get table of contents or index of a library * Combine libraries, control library page size, use switches for combinations, process complex library requests, use wildcards, and do library directives from command files * Complete with user's manual for only $29.95

Developer's Library

Contains the source code for for all library functions, including the transcedentals, memfiles, and those written in assembler. $25.00 with order, $50.00 if ordered later. (Sold only to Eco-C88 owners.)

Ecosoft Inc.
6413 N. College Ave.
Indianapolis, IN 46220

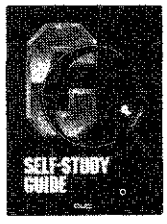
o-C88 C compiler requires an IBM PC, XT, or AT (or compatible) with 6K of memory, 2 disk drives and MSDOS 2.1 or later.



C Programming Guide

(Purdum, Que Corp.)

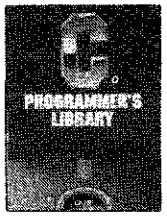
The second edition of the B. Dalton bestseller. Perfect for those just getting started with C. Includes discussion of many X3J11 ANSI Standards Committee recommendations. Many error messages from Eco-C88 give page references to this book. Price is $20.00 plus $2.00 shipping.



C Self-Study Guide

(Purdum, Que Corp.)

Using a question-answer approach, this book is filled with shortcuts, tips, techniques, and traps to avoid when learning C. Price is $17.00 plus $2.00 for shipping.



C Programmer's Library

(Purdum, Leslie, Stegemoller, Que Corp.)

Another B. Dalton bestseller. An intermediate C text for the programmer that wants to get the most from the language. Contains source code for many functions including an ISAM file handler. Price is $22.00 plus $2.00 for shipping.

To order, call or write:

1-800-952-0472 (for orders) or **1-317-255-6476** (tech. info.)

ORDER FORM CLIP & MAIL TO:
Ecosoft Inc., 6413 N. College Ave., Indianapolis, IN 46220

- ☐ Eco-C88 C Compiler $59.95 _____
- ☐ Eco-C88 Flexi-Graph Graphics Package $39.95 _____
- ☐ Eco-C88 Windowing Library $29.95 _____
- ☐ Ecosoft Librarian $29.95 _____
- ☐ Developer's Library $25.00 ($50.00 if not with order) _____
- ☐ C Programming Guide $20.00 _____
- ☐ C Self-Study Guide $17.00 _____
- ☐ C Programmer's Library $22.00 _____

SHIPPING $4.00 _____

TOTAL (IND. RES. ADD 5% TAX) _____

PAYMENT: ☐ VISA ☐ MC ☐ AE ☐ CHECK

CARD# _____ EXPIR. DATE _____

NAME _____

ADDRESS _____

CITY _____ STATE _____

ZIP _____ PHONE _____

Number 8 on the Reader Service Card

ECOSOFT

PRODUCED 2004 BY BMT MICRO
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal For IBM PC Programmers

PROGRAMMER'S Journal

Volume 5.3
May/June 1987

Contents

4 **Note from the Editor**

6 **Letters**

10 **PJ Forum:** Starting Your Own Software House *by Ray Duncan*
Ways to publicize your product

16 **Redirections:** Unix Books, Part 2 *by Frank D. Greco*
Frank Greco continues his survey of the literature on Unix

24 **Consulting:** Merger Mania *by Paul Barkley*
Why and how you should grow

30 **On Grapics:** EGA Data Control *by Michael Abrash*
More explanation of the EGA and a look at three EEGAs

38 **Modula 2:** Modula-2 for Pascal Lovers *by Tom Swan*
Will Modula-2 replace Pascal?

44 **Pascal:** Optimizing Turbo Pascal
Proceedings of the Borland Compuserve SIG seminar

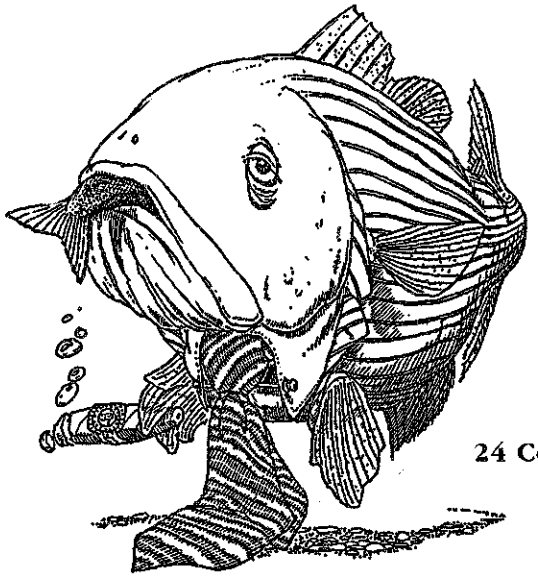
46 **C:** The Proposed ANSI C Language Standard, Part 7 *by Rex Jaeschke*
More on function prototypes

50 **C:** Coping with ANSI C and Forcing File Updates *by William James Hunt*
The standard and older version of the compiler; forcing DOS to flush data and control information

54 **BASIC:** Formatting Basic Output with Buffered PRINT USING *by Paul Dymerski*
Format text and numbers to a string in memory

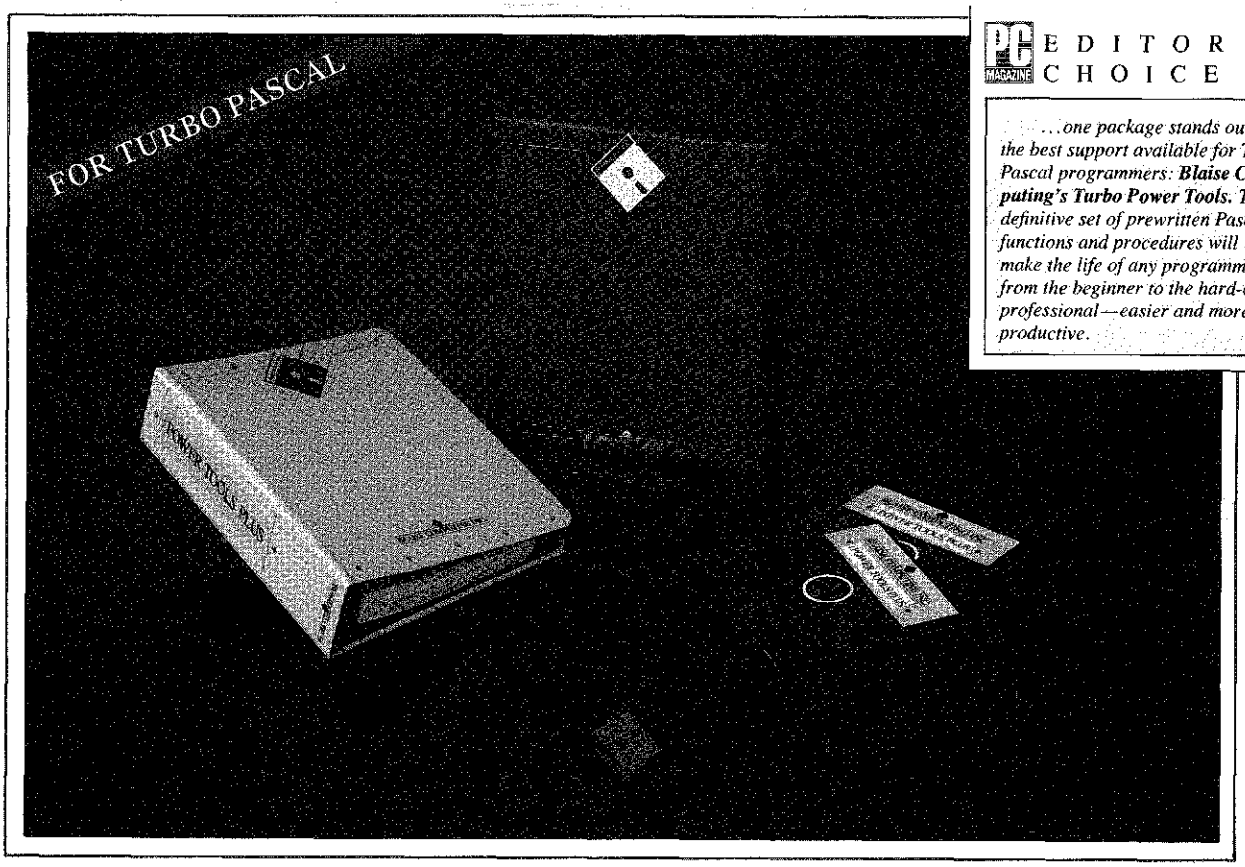
58 **Misc.**

64 **PJ Billboard**



24 Consulting

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



PC MAGAZINE EDITOR'S CHOICE

*...one package stands out as the best support available for Turbo Pascal programmers: **Blaise Computing's Turbo Power Tools.** This definitive set of prewritten Pascal functions and procedures will make the life of any programmer—from the beginner to the hard-core professional—easier and more productive.*

ANOTHER PLUS FROM BLAISE COMPUTING

The best just got better! Turbo POWER TOOLS, acclaimed as the best programmer support package for Turbo Pascal, now has even more functions, more detailed documentation and more sample programs.

Power Tools Plus™

Window Routines. Memory Resident Routines. Routinely.

have distinguished Blaise Computing over the years.

Turbo POWER TOOLS PLUS** supports Turbo Pascal Version 2.0 and later and is just **$99.95.

NO SECRETS

Turbo POWER TOOLS PLUS is crafted so that the source is efficient, readable and easy to modify. We don't keep secrets! We tell you exactly how windows are managed, how interrupt service routines can be written in Turbo Pascal, and how to write memory resident programs that can even access the disk. Maybe you've heard of some undocumented DOS features that resident programs use to weave their magic. Turbo POWER TOOLS PLUS documents these features and lets you make your own magic!

Here's just part of the PLUS in Turbo POWER TOOLS PLUS:

- **WINDOWS** that are stackable, removable, with optional borders and a cursor memory.
- **FAST DIRECT VIDEO ACCESS** for efficiency.
- **SCREEN HANDLING** including multiple monitor and EGA 43-line support.
- **POP-UP MENUS** which are flexible, efficient and easy to use, giving your applications that polished look.
- **INTERRUPT SERVICE ROUTINES** that can be written in Turbo Pascal without the need for assembly language or inline code.
- **INTERVENTION CODE** lets you develop memory resident applications that can take full advantage of DOS capabilities. With simple procedure calls, you can "schedule" a Turbo Pascal procedure to execute either when a "hot key" is pressed, or at a specified time.
- **PROGRAM CONTROL ROUTINES** allow you to run other programs from Turbo Pascal, and even execute DOS commands.
- **MEMORY MANAGEMENT** allows you to monitor, allocate and free DOS-controlled memory.
- **DIRECTORY AND FILE HANDLING** support to let you take advantage of the newer features of DOS including networking.
- **STRING** procedures allowing powerful translation and conversion capabilities.
- **FULL SOURCE CODE** for all included routines, sample programs and utilities.
- **DOCUMENTATION, TECHNICAL SUPPORT** and attention to detail that

*Another quality product from Blaise Computing: **Turbo ASYNCH PLUS™***

A new package which provides the crucial core of hardware interrupt support needed to build applications that communicate. ASYNCH PLUS offers simultaneous buffered input and output to both COM ports at speeds up to 9600 baud. The XON/XOFF protocol is supported. Now it also includes the "XMODEM" file-transfer protocol and support for Hayes compatible modems.

The underlying functions of Turbo ASYNCH PLUS are carefully crafted in assembler for efficiency and drive the UART and programmable interrupt controller chips directly. These functions, installed as a runtime resident system, require just 3.2K bytes. The high level function are all written in Turbo Pascal in the same style and format as Turbo POWER TOOLS PLUS. All source code is included for just $99.95.

BLAISE COMPUTING INC.

2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

ORDER TOLL-FREE 800-227-8087

Calif. residents call (415) 540-5441

YES, send me the PLUS I need! Enclosed is $_____ for ☐ Turbo POWER TOOLS PLUS ☐ Turbo ASYNCH PLUS ☐ OTHER _____. (CA residents add 6½% Sales Tax. All domestic orders add $10.00 for Federal Express shipping.)

Name: _____ Phone: (____) _____

Shipping Address: _____

City: _____ State: _____ Zip: _____

VISA or MC #: _____ Exp. Date: _____

Number 2 on the Reader Service Card

Reprinted from PC Magazine, June 10, 1986 Copyright © 1986 Ziff-Davis Publishing Company

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal For IBM PC Programmers
PROGRAMMER'S Journal

Editor
Robert Keller
Managing Editor
Erin O'Connor
Technical Editor
M. Steven Baker
Contributing Editors
Tom Swan, Frank D. Greco

Contributing Writers
Michael Abrash
Paul Barkley
Ray Duncan
Paul Dymerski
Frank D. Greco
William J. Hunt
Rex Jaeschke
Tom Swan

Publisher
Elizabeth Oakley
Production
Ralph Bentley
Advertising
Robert Tappan
Illustration
Jim Carpenter
Fulfillment
Gary Hoyer
Founder
Greg Estes

Business and Editorial Offices:
Programmer's Journal
29 W. 29th Ave., Suite 5
Eugene, OR 97405
(503) 484-2162

Change of Address:
Programmer's Journal
P. O. Box 3000
Dept. EE
Denville, NJ 07834
(505) 484-2162

Subscriptions:
1 year (6 issues) $24 U.S. ($30 in Canada and $40 elsewhere)
2 years $44 U.S. ($55 in Canada and $75 elsewhere)
3 years $64 U.S. ($80 in Canada and $110 elsewhere)

Articles and letters in Programmer's Journal represent the views of their authors and not necessarily those of the publisher. Letters to the editor become the property of Programmer's Journal and may be edited and published.

Advertising: For ad rates, deadlines and placement, contact Ralph Bentley, Robin Tappan or Gary Hoyer (503) 484-2162, or write to Programmer's Journal, P. O. Box 30160, Eugene, OR 97403.

IBM and Personal Computer are trademarks of International Business Machines. dBASE II and dBASE III are trademarks of Ashton-Tate.

Programmer's Journal (ISSN 0747-5861) is published bi-monthly by Oakley Publishing Company. Second-class postage paid at Eugene, OR, and additional mailing offices.

POSTMASTER: send address changes to Programmer's Journal, P. O. Box 3000, Dept. EE, Denville, NJ 07834

©1987 OAKLEY PUBLISHING COMPANY

NOTE FROM THE EDITOR

West Coast Computer Faire

I'm typing this on a PC in the press room at San Francisco's Moscone Center (a true-blue 4.77 MHz machine).

Mac Attack

I had a sobering talk with someone who's been marketing music software here for the last six years. Music software is a miniscule part of the market, and musicians are atypical, but what they feel might be shared to some degree by everyone.

The retailer said, "Hey, I love the PC. PC software is the first thing that I show when a customer comes in. It's fast, good, and it's what I use myself. But I can't tell you how many times I've had a person come in and even ask for a particular piece of PC software by name—he's heard of it, and wants to see it. I sit down with him, and I demo it, and often he'll just say, 'Yeah.' Then his eyes will begin to wander around the room until they settle on the Mac, and he'll say, 'Say, can you show me what you have on that?'

"I don't think we've figured out why yet. We're always debating it. Is it the display? Is it the mouse? Is it the Apple image?" "Is it the portability?" I asked. "Naw," he said, "I don't think so. In any case, Mac software is outselling PC software by a wide margin." I think the retailer was dramatizing a bit for the benefit of PC software developers, but it made an impression on me.

Displays at the Faire

That brings me to displays. AT&T is showing its new Truevision capture and display board, which is driven by a TI 34010 and gives 16 million colors at 1Kx1K, 32K colors at 2Kx1K. Bring $5995 if you want one.

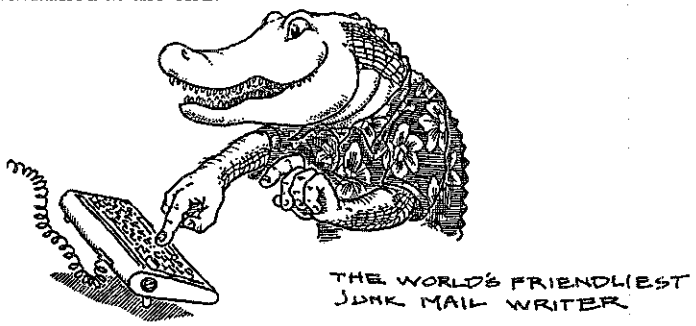
PC Pro in Pacifica has a booth with Ventura running on both a Wyse WY-700 monitor and a Sigma Designs 19" monochrome monitor ($2995). Both beautifully display a full-page layout.

The Amiga and Atari graphics here are eye-catching. Most *PJ* readers have probably seen them—smooth animations of juggling chrome balls reflecting complex, multi-colored, moving scenes. (Most *PJ* readers are also aware of the amount of work involved in such demos.)

The First Amiga Users' Group is showing a pretty hi-res color digitizing camera and paint program with which a salesman planted a realistic-looking third eye in the middle of a model's forehead.

Someone from Apple gave a short demo of the Mac II's sound and graphics capabilities. The sound was afflicted with the kind of digital aliasing that has been described as "jingling keys." Once I notice that sound, I can't get it out of my head—piano with jingling keys, flute with jingling keys. The display looked good, though.

David Bunnell gave an interesting little talk in which he revealed his origins as a disaffected high school teacher, and Steve Wozniak talked with undiminished zeal about Apple's beginnings. "Woz, you've gotta work on your enthusiasm," John Dvorak remarked at the end.



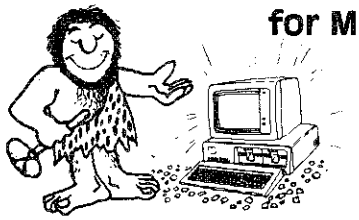
Jim Carpenter did the drawing of Hal Hardenbergh above. Hal suggested the portrayal. Hal isn't in this issue, absent for the first time in a long while. If you're curious about what he writes, look at any of the last year and a half's issues.

—Robert □

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Professional Programming Products

for Microsoft C, PASCAL, FORTRAN, and Assembly Language



FREE!

PC-WRITE™ text editor

PROGRAMMERS AND SOFTWARE DEVELOPERS - LOOK AT THESE PRODUCTS!
NO ROYALTIES REQUIRED

ASMLIB
The Programmer's Library

- A linkable library with over 210 assembly language subroutines which are directly CALLable from Microsoft™ Assembler, C, PASCAL, and FORTRAN.
- Virtual disk file handling.
- Text WINDOWing allows up to 64 overlapping windows.
- Int. driven asynch. support.
- Graphics on EGA, herc. and CGA.
- Floating point math and trig outlines with 8087 support.
- Installable keyboard activated programs are easily written with ASMLIB's special functions.
- Plus much, much more.
- Supplied with complete source code.

Only $149.00 Complete

asmTREE
The Programmer's B+Tree Data File Management System

- A complete single/multiuser database development system written in assembly language and CALLable directly from Microsoft™ Assembler, C, FORTRAN, PASCAL, and LATTICE™ C programs.
- Up to 256 users.
- Up to 256 index and data files.
- Multiple key types.
- Multiple indices per index file.
- Duplicate and variable length keys.
- Read DBASE data files directly.
- Plus much, much more
- Supplied with complete source code.

Only $395.00 Complete

NEW

FPLIB
FPLIB™-IEEE Floating Point for REALIA™ COBOL!

- A full IEEE floating point math package for your REALIA COBOL programs.
- 8087 numeric co-processor support and EMULATION. Switching from emulation to true 8087 instructions is done automatically at RUN time - transparent to the user!
- A full set of both arithmetic and TRIGONOMETRY functions have been provided.
- Conversion routines allow your REALIA program to pass data to and from FPLIB by either using DISPLAY types or COMP-5 data types.
- FREE Updates for 1 year.
- SOURCE CODE available.

Only $149.00 Complete

NEW

Turbo.ASM
For TURBO Users!
Turbo.ASM™ (turbo dot a.s.m.)

- A unique programming tool which is a must for every Turbo Pascal user. The only package designed for the Turbo user who wants to get into Assembly Language.
- Outlines 4 different ways to add assembly language routines to Turbo programs without effecting your code space.
- Fully explains internal workings of Turbo Pascal and data passing methods.
- Includes a library of Assembly routines which can be used directly from Turbo Pascal.
- NO ROYALTIES required.

Only $99.95 Complete

B C ASSOCIATES
3261 No. Harbor Blvd., Suite B
Fullerton, CA 92635
1-800-262-8010
in Calif. Call
(714) 526-5151

Call for information on complete line of SimpleNET networking products.

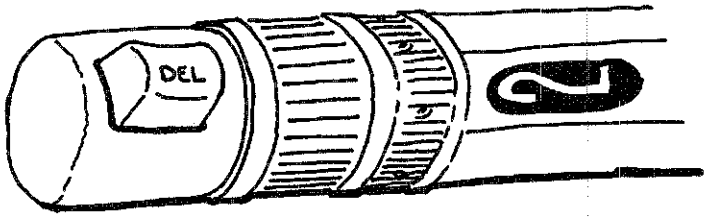
Outside CA, call TOLL FREE 1-800-262-8010

USE YOUR VISA OR MASTERCHARGE -

All prices include UPS shipping within continental United States. Outside U.S. please add $10 per package. Calif. residents please add 6.5% sales tax.

Number 36 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Advanced Trace86 Addendum

I'm writing to point out an undocumented feature in Version 2.3 of Advanced Trace86, which Peter Feldmann reviewed in the November/December 1986, issue of *Programmer's Journal* (Volume 4.6).

According to Sargent and Shoemaker's Revised Edition of *The IBM PC From the Inside Out*, Scroll Systems's SST debugger, essentially the same product as AT86, has a DISK command useful for snooping of various kinds. When the DISK command is given, AT86 loads disk sectors, starting at the boot sector, into memory, and a display similar to that of the memory display/modify function is shown.

Similar Ctrl-[key] commands are available for overtyping, toggling between hex, ASCII, and byte format displays, and for moving around the disk. Typing a ? brings up a help screen describing the various functions available; for example, Ctrl-F displays the FAT and even indicates the cluster chaining. Ctrl-P displays the root directory, with subdirectories displayed by Ctrl-D (the general display cluster command, valid also for showing data clusters of directory or FAT entries) while the cursor is on the subdirectory entry.

DOS relative sector numbers are easily obtained and can be used with the LOAD and WRITE commands as indicated in the documentation. On the other hand, disks, both floppy and hard, are easily rendered useless by unintended commands. When AT86's disk functions are used along with the capabilities of Morgan Computing's elegant Disk Toolkit, magnetic media become a great deal of fun.

AT86 works very well with the 43-line mode of the EGA and can be used with dual monitors to great advantage. Tracing a program on one screen while the output goes to another is a fascinating adventure.

Gary Darsey
Tucson, AZ 85730

California and Texas

Cheers on getting Ray Duncan to unburden himself on the ins and outs of starting a software company. He notes, of course, that much of what he says applies to starting up a software house in his own state of California. We seem to have a better business climate here in Texas (I know we do—I lived in California for eight years). Items:

1. In Texas, the company name is registered with the county clerk as an "assumed name certificate." Your bank can handle the whole thing for you when you set up your account, and there is no fee.
2. Texas banks may improve now that the banking law has been changed. Almost all small banks are satelites of large holding companies, and the local staffs don't even have control over the bookkeeping. The main bank offices are too large to have any interest in small businesses. If you need to borrow money, the best bet is to have a lot of out-of-state credit cards (the in-state cards are from companies with credit lines too small for this type of use). The staff in the bank doesn't have the least idea what is required for the operation of a small business.

   The other problems are probably temporary: Texas has a law that says you can't borrow money with your house as collateral. Texas S&Ls and banks are presently overflowing with foreclosed property and are lending very little money right now.
3. Until you have a business *location* (an office in an area zoned for business), no licenses of any kind are required in this county (Tarrant) or city (Arlington). It's actually illegal (per the city) to operate a business out of your home (if it's in an area zoned only for residences), but only a complaint from your neighbors would trigger enforcement of this law. Unless customers are coming in all the time and trucks are backing up to be loaded, neighbors will not complain about the UPS truck stopping once a day as it stops occasionally at their houses, too.

   You will have to get a tax number from the state. An 800 number listed in the 'phone book gets you in touch with the tax people, and they send out a packet of forms to be filled in. You have certain requirements for collecting sales tax on sales within the state.
4. I don't understand why Mr. Duncan says that you need a "Federal Employer ID Number." I have done business via purchase order with companies all over the US for the past three years, and none has ever withheld any part of what they owe me. Some *do* report amounts of $500 or over to the IRS (they are required to do so by law, but some are lax in this). They *never* ask for the tax number until they are closing their books at the end of the year, and my Social Security number has always sufficed. Nevertheless, it costs little effort to do as Mr. Duncan suggests.
5. I don't know of any deposit required against the sales tax in Texas. It is advisable to set up a savings account in which to put the sales tax money so that you will collect interest on it while it is being held. I don't have a lot of experience with these folks (most of my business is either not subject to sales tax or is out-of-state), but I don't think they are as difficult to deal with as California's State Bureau of Equalization (I *do* have experience with them).
6. There is no income tax in the state of Texas, and I think there would be a popular revolution if the legislature passed one. Any governor who signed the bill would probably be impeached and would certainly not be elected again. (As you might imagine, Texas government is running deep in the red since the oil business went away.)
7. Getting started with UPS here required no deposit and no volume estimate. They do expect to get paid for each shipment and levy a small weekly service charge ($3.25 for the first of the week).

Apart from these rather minor items of difference from state-to-state, I found Mr. Duncan's article "right on." It's always helpful to find out how others are handling things, and neophytes won't have any of this information (I sure didn't when I started).

William A. Manly
HMI Consulting
Arlington, TX 76016

A Call Procedure

I read with interest Tom Swan's article, "Teach an Old Compiler a New Trick," in the July/August 1986, issue of your magazine. I also read Roderick K. Graves's letter in the November/December 1986, issue, showing a slightly faster way to implement the CALL procedure. Since I had just independently implemented my own CALL procedure, the article and letter were of special interest to me. I am somewhat perplexed that both Tom and Roderick worked so hard to defeat the work that the compiler had done by removing unneeded words from the stack. If you let a procedure terminate normally, the compiler code takes care of adjusting the stack. I implemented my CALL procedure as shown below.

```
type PROC = integer;

procedure call(Process: integer);
begin
  inline($ff/$56/$04);    { CALL [BP+4] }
end;
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

POWER TOOLS for SYSTEM BUILDERS™

1-800-543-6277
Ask for Operator 2053
California 1-800-368-7600

TSF is owned and operated by programmers, so we understand your needs. We believe and practice integrity in all business dealings. We advertise real prices and offer the best possible terms to all customers. We provide prompt delivery of current product versions.

We carry only products which we have personally used or which come with strong recommendations from developers we respect. We have the technical capability and the interest to search for and find new products to meet your specific needs. We will gladly provide quotes for volume purchases or for products outside our normal product line. We consider it our job to help you get answers when publisher's technical support is unresponsive. At TSF service means more than a pushy sales pitch and a $2 lower price.

We accept checks, Visa, Mastercard, American Express and COD. We charge your card only when we ship and do **not** add a surcharge. We provide free UPS delivery on software orders over $100 ($3 delivery charge on orders under $100). Our COD fee is $4. We allow return privileges on most products. Give us a try.

The Software Family

649 Mission Street
San Francisco, CA 94105
(415) 957-0111
1-800-543-6277 (U.S.)
1-800-368-7600 (Calif.)
Ask for Operator 2053 on toll free calls

New! Supercharge Turbo Pascal

Mach2 from MicroHelp enhances Turbo Pascal with instant screen updates, windows, Ctrl-Break and DOS critical error trapping, flexible scrolling, Lotus-style menus and much more. Most functions are written in assembler for lightning fast operation and low memory overhead. Mach2 gives Turbo Pascal programs the i/o performance and hardware error handling capabilities of well-written assembly language programs. No royalties. All source included. (List $69) Only $55 from TSF. 30 day money back guarantee.

New! Create Demos AND Applications

Screen Machine from MicroHelp lets you paint screens to create demonstration prototypes for selling and refining your program concept. Any PC screen can be captured as a starting point or you can work from scratch. After finalize the concept, Screen Machine generates screen display code in Basic, Turbo Pascal, dBase II or III, or assembler (including support for Turbo Pascal and Basic Mach2 if desired). No royalties. All source included. (List $79) Only $63 from TSF. 30 day money back guarantee.

New! Visual Command Language

Opal from The Software Factory lets you overcome MSDOS batch limitations. Run any sequence of MSDOS programs using control logic based on the system date and time, operator entry (including comprehensive prompting and editing), the existence of files or the contents of files. A comprehensive but easy to use screen painter allows you to quickly generate menus providing cursor-key item selection with full control of display attributes. Complete with screen painter. Opal provides a cost effective alternative to custom written shells. Only $79. 30 day money back guarantee.

New! Professional Hardware Diagnostics

Almost every PC user has wasted time and money due to mis-diagnosis, unnecessary hardware service calls or technicians arriving without necessary parts. Windsor Technologies now provides comprehensive, menu driven diagnosis software with detailed problem reports in English. With the Windsor tools you can easily identify failed components and make sure that service technicians are called only when needed and that they know in advance what parts to bring. The service kits are used many national repair companies, work for IBM, clone and add-on components and inlcude terminators for loop-back testing of serial and parallel ports. The Windsor diagnostics let you provide the hardware expertise clients expect from you. Model T01 for PC/XT $195. Model T51 for AT $245. Model T91 for PC/XT/AT $395. 30 day money back guarantee.

Send Or Call For Comprehensive Free Catalog

TSF carries a complete line of software and hardware tools for programmers and system engineers. We offer "big name" products at competitive prices and a large selection of hard-to-find products that save you hours of research, evaluation and development work. Our catalog includes hype-less product descriptions that explain what our products do to improve your productity and your product's quality. Call or send the coupon for your free copy.

__Please send me a free catalog

__Please send the following products:
_____

Name: _____
Address: _____
City/State/Zip: _____

Credit Card: _____
Exp: _____

Mail to: TSF, 649 Mission Street, San Francisco, CA 94105

Number 98 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The drawback to this method is that you need somewhat more stack area. Nested CALLs cause the stack to get larger slightly faster. Its obvious advantages are clarity and speed. Tom's method takes 92 machine cycles to get back to the procedure that did the CALL, and Roderick's takes 81 cycles. My CALL takes 74 machine cycles.

I use this CALL procedure as the basic part of a table-driven finite state machine. Since there is quite a bit of overhead involved with the finite state machine, any speed gains are welcome.

One final thought: While it is true that the CALL procedure may be used anywhere in a program without worrying about forward references, the initialization of the process parameter used in the CALL procedure must be done forward of the procedure to which it is being made to point.

Michael Burton
Key Tronic Corporation
Spokane, WA

Execution Times

I read your magazine with great interest, and in the past few months the problem of determining exact execution times of code has arisen several times.

I was persuaded by Mr. Hoyt's analysis of the problem. In the "early days," designers "bundled" the fetch times for an instruction with its execution, and one could calculate very accurately the execution time of any string of code. For such machines, Mr. Hoyt's methodologies were absolutely correct.

Then Mr. Abrash added a notable contribution when he showed that fetch times and the size of the pre-fetch queue have become an essential ingredient in such an analysis.

In his recent article on EGA adapters, Mr. Abrash asked if there might be a conclusive method of determining actual execution time with precision. This letter offers a methodology that I have employed with success since the late '60s.

The method is based on executing a section of code 1,000 times, or 10,000 times, or 100,000 times, or a million times. The total time can be divided by the number of executions, and the resulting time for a single execution can be determined within a small sigma (or standard deviation).

Start out by setting up a loop, including those mechanisms for timing the execution of the loop, whether it be by putting out messages for you to use while manipulating a stop-watch, or by reading the system clock. Disable interrupts if you need to (again, remember that timing is your responsibility if you do, since the system clock won't work).

Once you have set up the loop and timed it, add the code that you are examining. The difference between the first time and the second time divided by the repetitions gives you the time for a single execution of the code you are testing.

Just beware of caches between the size of the code that you are examining and the size of your total measurement code.

Well, Mssrs. Hoyt and Abrash, keep your work coming to us. I hope the above will help repay you for the pleasure and instruction I have received from your work.

Thomas B. Reifsnyder
Orange, CT 06477

Readers are referred to the excellent treatment of this subject by Byron Sheppard in the January '87 BYTE. *Byron's approach is to use a channel of the system timer to time durations as low as 840 ns, and he then times just one iteration of his test code. Byron points out, in an anecdotal way, the large variations in elapsed time introduced by the state of the pre-fetch queue at the start of the test section, illustrated by the difference in execution time when the code is called (which dumps the queue) and when it is run in-line (which doesn't necessarily start with an empty queue). Perhaps a further step in evaluating a piece of code is to systematically return the time for the best- and worst-case queue situations. That could be done during either a single-pass or loop timing by padding the front of the code with dummy code guaranteed to either drain the queue or leave it full.*

Thomas Reifsnyder touches on another consideration as well—what interrupts are taking place, and what effect will they have on execution?

These hairs needn't be split unless necessary, but for certain applications, like the one I'm working on right now, it's crucial.—Editor □

TAKE COMMAND.

INTRODUCING THE SURE-FIRE WAY TO TAKE COMMAND OF MS-DOS

INTRODUCING COMMAND PLUS ONLY $79.95

Now you can take command of MS-DOS with Command Plus, the programming shell and command processor that replaces COMMAND.COM in Versions 2.0 and above.

What this means is that you can now program efficiently in DOS without having to learn UNIX.

Programming efficiently means that Command Plus will increase your programming productivity and flexibility by virtually eliminating the repetitive tasks DOS COMMAND requires.

For example, Command Plus' aliasing function lets you create fast, memory resident macros, while the history command lets you use the cursor keys to recall, edit and execute commands that you ran up to 48 command lines ago.

What's more, Command Plus also equips you with Script, a batch processor that's easy to learn and unbelievably powerful. It's Pascal-like language includes control loops, conditionals and variables which let you create unique systems utilities.

In short, Command Plus means that you can now program easily and efficiently in the familiar DOS without having to learn UNIX.

It's no wonder that Ray Duncan has already called Command Plus "a very well designed and sound product."

And has written that "The regular expression support, the enhanced copy, move and del commands, and the browse utility are particularly neat."

At only $79.95, plus shipping and handling, the price of Command Plus is pretty neat, too.

So, if you want a sure-fire way to take command of DOS, or if you want more information about Command Plus, call us at (800) 992-4ESP. In California, call (213) 390-7408. VISA and MasterCard accepted.

MS-DOS is a registered trademark of Microsoft Corporation.

(800)992-4ESP (213)390-7408

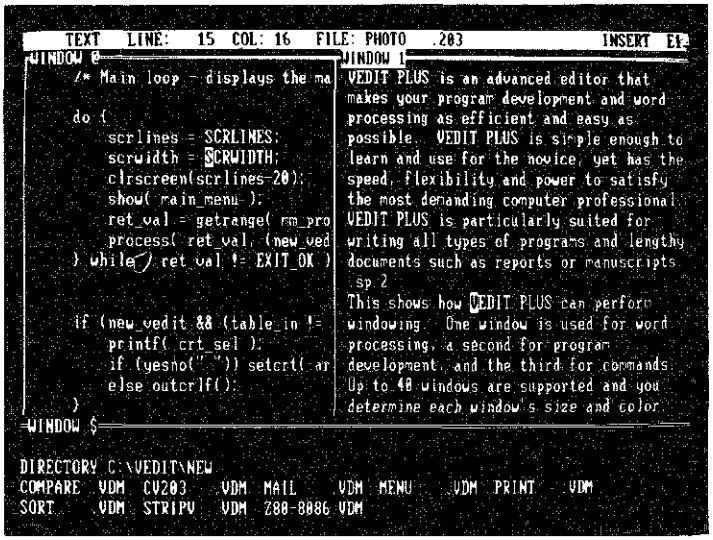
COMMAND PLUS.

ESP SOFTWARE SYSTEMS INC.
11965 Venice Boulevard
Suite 309
Los Angeles, CA 90066

ESP

Number 111 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED


TEXT LINE: 15 COL: 16 FILE: PHOTO .203 INSERT
WINDOW @
/* Main loop - displays the ma
do {
scrlines = SCRLINES;
scrwidth = SCRWIDTH;
clrscreen(scrlines-20);
show(main_menu);
ret_val = getrange(rm_pro
process(ret_val, (new_ved
} while(ret_val != EXIT_OK)
if (new_vedit && (table_in !=
printf(crt_sel);
if (yesno(".")) setcrt(ar
else outcrlf();
}
WINDOW $
WINDOW 1
VEDIT PLUS is an advanced editor that makes your program development and word processing as efficient and easy as possible. VEDIT PLUS is simple enough to learn and use for the novice, yet has the speed, flexibility and power to satisfy the most demanding computer professional. VEDIT PLUS is particularly suited for writing all types of programs and lengthy documents such as reports or manuscripts.
sp 2
This shows how VEDIT PLUS can perform windowing. One window is used for word processing, a second for program development, and the third for commands. Up to 40 windows are supported and you determine each window's size and color.
DIRECTORY C:\VEDIT\NEW
COMPARE .VDM CV203 .VDM MAIL .VDM MENU .VDM PRINT .VDM
SORT .VDM STRIPV .VDM Z80-8086 VDM

VEDIT PLUS

#1 PROGRAMMABLE EDITOR

(Call for FREE DEMO disk)

Stunning speed. Unmatched performance. Total flexibility. Simple and intuitive operation. The newest VEDIT PLUS defies comparison.

Try A Dazzling Demo Yourself.

The free demo disk is fully functional - you can try all features yourself. Best, the demo includes a dazzling menu-driven tutorial - you experiment in one window while another gives instructions.

The powerful 'macro' programming language helps you eliminate repetitive editing tasks. The impressive demo/tutorial is written entirely as a 'macro' - it shows that no other editor's 'macro' language even comes close.

Go ahead. Call for your free demo today. You'll see why VEDIT PLUS has been the #1 choice of programmers, writers and engineers since 1980.

Available for IBM PC, TI Professional, Tandy 2000, DEC Rainbow, Wang PC, MS-DOS, CP/M-86 and CP/M-80. (Yes! We support windows on most CRT terminals, including CRT's connected to an IBM PC.) Order direct or from your dealer. $185.

Compare features and speed	BRIEF	Norton Editor	PMATE	VEDIT PLUS
Off the cuff' macros	No	No	Yes	Yes
Built-in macros	Yes	No	Yes	Yes
Keystroke macros	Only 1	No	No	100 +
Multiple file editing	20 +	2	No	20 +
Windows	20 +	2	No	20 +
Macro execution window	No	No	No	Yes
Trace & Breakpoint macros	No	No	Yes	Yes
Execute DOS commands	Yes	Yes	Yes	Yes
Configurable keyboard Layout	Hard	No	Hard	Easy
'Cut and paste' buffers	1	1	1	36
Undo line changes	Yes	No	No	Yes
Paragraph justification	No	No	No	Yes
On-line calculator	No	No	No	Yes
Manual size / index	250/No	42/No	469/Yes	380/Yes
Benchmarks in 120K File:				
2000 replacements	1:15 min.	34 sec.	1:07 min.	6 sec.
Pattern matching search	20 sec.	Cannot	Cannot	2 sec.
Pattern matching replace	2:40 min.	Cannot	Cannot	11 sec.

VEDIT PLUS FEATURES

- Simultaneousy edit up to 37 files of unlimited size.
- Split the screen into variable sized windows.
- 'Virtual' disk buffering simplifies editing of large files.
- Memory management supports up to 640K.
- Execute DOS commands or other programs.
- MS-DOS pathname support.
- Horizontal scrolling - edit long lines.
- Flexible 'cut and paste' with 36 'scratch-pad' buffers.
- Customization - determine your own keyboard layout, create your own editing functions, support any screen size.
- Optimized for IBM PC/XT/AT. Color windows. 43 line EGA.

EASY TO USE

- Interactive on-line help is user changeable and expandable.
- On-line integer calculator (also algebraic expressions).
- Single key search and global or selective replace.
- Pop-up menus for easy access to many editing functions.
- Keystroke macros speed editing, 'hot keys' for menu functions.

FOR PROGRAMMERS

- Automatic Indent/Undent for 'C', PL/I, PASCAL, etc.
- Match/check nested parentheses, i.e. '{' and '}' for 'C'.
- Automatic conversion to upper case for assembly language labels, opcodes, operands with comments unchanged.
- Optional 8080 to 8086 source code translator.

FOR WRITERS

- Word Wrap and paragraph formatting at adjustable margins.
- Right margin justification.
- Support foreign, graphic and special characters.
- Convert to/from WordStar and mainframe files.
- Print any portion of file; selectable printer margins.

MACRO PROGRAMMING LANGUAGE

- 'If-then-else', looping, testing, branching, user prompts, keyboard input, 17 bit algebraic expressions, variables.
- Flexible windowing - forms entry, select size, color, etc.
- Simplifies complex text processing, formatting, conversions and translations.
- Complete TECO capability.
- Free macros: • Full screen file compare/merge • Sort mailing lists • Print Formatter • Menu-driven tutorial

VEDIT and CompuView are registered trademarks of CompuView Products, Inc. BRIEF is a trademark of UnderWare, Inc. PMATE is a trademark of Phoenix Technologies Ltd. Norton Editor is a trademark of Peter Norton Computing Inc.

CompuView

1955 Pauline Blvd., Ann Arbor, MI 48103 (313) 996-1299, TELEX 701821

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Starting Your Own Software House: Getting the Word Out

by
Ray Duncan

This is the second in a series of articles directed at the programmer who wants to start up a software house of his or her own. I am not addressing intrinsic problems of designing, coding, and testing a new software product in this series—such matters are second nature to any subscriber to *Programmer's Journal*! Instead, my focus is on the mundane but vital problems of establishing and maintaining a viable small business on your way to becoming a software mogul. In last month's installment, I discussed preliminary issues: business licenses, bank accounts, shipping arrangements, and obtaining expert advice on legal and accounting matters. This month, we will ponder ways to make the buying public aware of your new brainchild.

Methods of publicizing your product can be grouped under the general headings:

- Reviews
- New product announcements/Press releases
- Direct mail
- Advertising

In typically perverse fashion, we will discuss these methods in order from most expensive and least effective to least expensive and most effective.

Advertising

Advertising will be your biggest expense aside from rent and payroll, and lack of prudence in placing your ads can make or break your fledgling company. Getting an advertisement into print is more work than you might think; it involves:

- choosing your advertising vehicles (magazines)
- placing the advertisement insertion order
- design and typesetting
- delivering the advertisement to the magazine before deadline
- checking the ad placement
- tracking the response to the advertisement
- paying the bill (the most painful part)

Choosing your media: Select the magazines to carry your advertisements strictly on the basis of their ability to reach potential customers for your product. Until like Lotus, Microsoft, or Compaq, you have millions to squander on "image-building" advertisements, you are only pouring your money down the drain if you put ads into magazines that are not read by the people who are likely to need your program. Similarly, you don't want to waste your money by placing identical advertisements in different magazines that have a large overlap in readership. Don't pick a magazine for its reputation or design; pick it after a cold-hearted assessment of what that magazine can do for you.

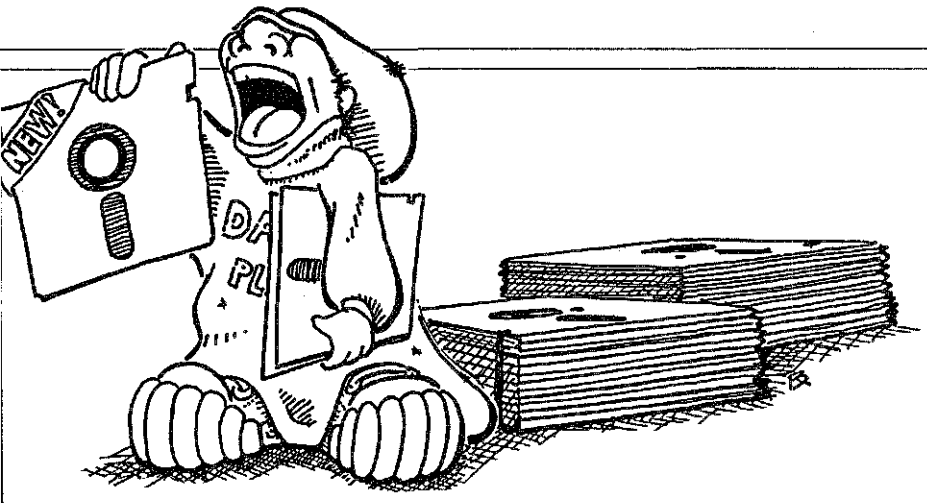
If your product is a programming tool for the IBM PC family of machines, there is a fairly small and easily identified set of magazines to be concerned with: *BYTE*, *Dr. Dobb's Journal*, *PC Tech Journal*, *Computer Language*, and *Programmer's Journal*. If you are writing a mass-market application rather than a programming tool, the main players in the IBM PC market are *PC Magazine*, *PC World*, *PC Tech Journal* again, *PC Week*, and *InfoWorld*. I include *PC Tech Journal* in the latter category because it is read by many semi-technically knowledgeable managers who want to keep up with new products.

If your product tends towards the hobbyist, you may want to look into magazines such as *Popular Computing* or *Compute!*, and such low-priced advertising vehicles as *Computer Shopper* or *Micro Times*. If your product is hardware oriented, for example a CPU emulator or a hardware debugging tool, you may want to explore the periodicals directed at working engineers—*Electronic Design*, *EDN*, *Mini-Micro Systems*, *Computer Systems News*, *EE Times*, *IEEE Micro*, and *IEEE Computer*. If you are oriented toward teaching and other academic pursuits, investigate *IEEE Software*, *Abacus*, and the various ACM journals.

Each magazine has a characteristic visual presentation, editorial style, and perceived reader. If I handed you a copy of *Programmer's Journal* and asked you what the focus of the magazine is, you might not be able to verbalize it. On the other hand, if you were to lay a copy of *Programmer's Journal* down next to a copy of *PC Tech Journal* and page through them simultaneously, comparing the style of graphics, typesetting, the tone of the articles and columns, and the kind of advertisers found within the pages, you would immediately agree that each magazine has a distinct personality. Readers who are not in tune with the editorial slant select themselves out by non-renewal, or by voting with their wallets at the newsstand.

If you are writing a program for a market with which you are very familiar, you already have the information in your head

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



that will enable you to pick the proper magazines for your advertisements. You know where your competitors advertise, you know where articles appear on topics related to your program, and you know where reviews of your competitors' products appear. Unless you are writing a totally new, unique program that simply has no competitors, you can't do much better than to go where the action already is. A little armchair psychology, and trust in your own intuition and common sense, will serve you well here.

Be conscious of subscriber overlap among your chosen magazines. Consider the members of the first category of publications we mentioned above: *BYTE*, *Dr. Dobb's Journal*, *PC Tech Journal*, *Computer Language*, and *Programmer's Journal*. *BYTE* has a circulation of some 400,000 and is read (or at least glanced through) by nearly everyone who buys one of the other four magazines. *PC Tech Journal* has the next largest circulation, recently passing 100,000. *Computer Language* and *Dr. Dobb's Journal* have circulations in the neighborhood of 40,000–60,000, while *Programmer's Journal* (I am guessing) has a circulation in the vicinity of 10,000–15,000. I suspect that the overlap in readers between *PC Tech Journal* and *Dr. Dobb's* is fairly large, and that the overlap between *Dr. Dobb's* and *Computer Language* is likewise large, but that the overlap between *PC Tech Journal* and *Computer Language* or *Programmer's Journal* is quite small.

While you try to choose your targeted victims (in terms of magazines to advertise in), the magazines are also trying to victimize you. Beware of ad salesmen! You might be friends with, or have trust in, a magazine's editorial staff, but believe me, the advertising staff is autonomous from the editorial staff, and their statements must be treated with great caution. Their incomes are usually based on commissions, and some of them will say anything to convince you to place an advertisement—especially if they sense your lack of experience.

Let me tell you a little parable. During the first year I was in business, I was called by a salesman for a now-defunct Wayne Green magazine, who stated that my product was receiving an extensive and glowing review in an upcoming issue. He waxed eloquent about what a smart move it would be to buy a full page advertisement in the same issue, which he would personally insure was printed facing the review. Being but a babe in the woods in these matters, I readily agreed. When the magazine appeared, it became clear that I had been had. My program was only mentioned in passing in a sentence with three or four other products. Now I must presume that practices as sleazy as this are not very common, but you have been warned!

Placing your ad insertion order: After you've identified the magazine(s) you would like to advertise in, look on the magazine's masthead for the phone number of the advertising department, call them up, and request a "media kit." This packet contains a recent copy of the magazine, a "rate card" giving the cost breakdown for each size of ad and any applicable surcharges for color and position, a list of ad order closing dates and materials deadlines, a sample contract, and various brochures giving the magazine's terms and conditions laced with a plentiful amount of sales hype.

Magazine	1/4 page black&white	full page black&white	full page 4-color	6-times discount	12-times discount
Byte	$3,280	$9,790	$11,170	5%	10%
DDJ	980	2,870	3,470	10%	20%
Computer Language	680	1,950	2,575	6%	10%
Programmer's Journal	325	995	1,545	10%	N/A (bimonthly)

Figure 1. Comparison of ad rates for some typical computer magazines, as of March 1, 1987. Additional surcharges may apply for cover positions or other preferred positions. These rates were in effect at the time this article was written but may have changed (upward, naturally) by the time you read this. If you supply the magazine with camera-ready copy, they will usually give you an additional 15% "agency discount" from the quoted prices.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The cost of advertising in a particular magazine is closely linked to that magazine's circulation (see Figure 1). At the high end, a one-time advertisement insertion in *BYTE* ranges from $3,280 for a quarter page black and white advertisement to $11,170 for a full page four-color advertisement. Additional charges may apply if you request a preferred position; for example, *BYTE* adds a 10% surcharge for a position "toward the front, right-hand page, outside," and a *BYTE* four-color back cover costs $13,935 per issue and can be obtained only with a 12-times contract. On the other hand, rates are substantially lower if you sign a multiple-insertion contract that commits you to place a certain number of ads within a one-year period. Although the language of these contracts may look a little scary, there is no real risk involved. If you fail to make the contracted number of insertions within the contract period, the magazine will simply back-bill you for any applicable difference in rates on the ads already inserted and paid for.

In the heyday of the PC boom (1982–1984), magazines were quite lenient about credit terms in order to sign up new advertisers. It was easy for fledgling software vendors to get committed to an advertising program they couldn't possibly afford and build up a tremendous debt. Then the poor vendor was really over a barrel: The magazines wouldn't accept additional ads until the old ones were paid for, and the software vendor couldn't afford to pay for the old ads unless the new ads appeared to keep the orders coming in. What a rat race! Of course, the magazines paid dearly for their credit leniency as droves of software companies and book publishers (particularly those who invested too deeply in the PCjr) folded or merged during the Great Slowdown.

Nowadays, most magazines are very strict about credit and will make you pay for all your advertisements in advance until you have shown yourself to be a stable citizen. Although inconvenient, this is probably safer for you anyway during the startup period. It keeps you from ordering ads you can't afford and forces you to keep much closer tabs on the relationship between ad expenses and ad results.

Design and production: The first step in putting together your advertisement is to sit down and think about your product from the consumer's point of view. What are the points that will grab his or her attention and make your product seem special? It behooves you to scrutinize the advertisements of your competitors, noting what they feel to be important, and perhaps, what they fail to emphasize. You might as well benefit from the dollars they have already spent on market research and ad design. When deciding what to put into your advertisement, remember the adage: "Stress benefits, not features." Readers don't want to see a shopping list of your product's little technical virtues. They want to know how they will profit by choosing your product over your competitors'.

Once you've settled on the contents of your advertisement, you need to meld all the pieces (text, pictures, other graphics, your company logo) together somehow into camera-ready copy that can be sent off to the magazine. There are many roads you can take to this goal.

First we can consider (and I hope, discard) the ultra-cheapo route: using MacPaint, Fancy Font, Gem Draw, or other programs of that ilk to design and produce camera ready copy on a laser printer. In the first place, this approach is practical only if your advertisement does not contain photographs or art other than line drawings. In the second place, such ads are easy to pick out, and they look awful compared to true typeset text. Laser printers such as the Hewlett Packard LaserJet simply don't have the necessary resolution.

A preferable but still inexpensive alternative is to make a mockup of your advertisement, either by typing and annotating it or by producing a rough facsimile on your laser printer, and then take it to a typesetter. Typesetting shops tend to be rather small operations that operate in conjunction with a specific printer or in the general area of several print shops. Although you have probably been completely unaware of them in the past, you will be amazed at the number of typesetters you will find within a mile or two of your office when you start looking for them. They are listed in the Yellow Pages under "Graphics Design" or "Typesetting." Typesetting and pasting up an average advertisement will cost you $100–$200. The turnaround time is usually one to three days. Proofread the result carefully before you pay for it; typesetters are often unfamiliar with computer terms. Corrections that must be made due to typesetting errors should be free while changes that are necessary because you changed your mind, or had errors in your original copy, will naturally cost you extra. It's a good idea to take someone else along to proofread the advertisement. Since you wrote the copy, your mind is predisposed to see what is supposed to be on the paper, and it is all too easy to overlook misspellings or missing or duplicated words.

An intermediate ground between home-brew and agency ad work is to get help from a freelancer: typically a struggling artist or graphics designer who does a little ad business on the side to make enough money to eat. Such a person will help you design the advertisement, arrange the typesetting, go over the final result with you and incorporate any changes that need to be made, and then deliver the finished product back to you ready for shipment to the magazine. In my experience, a freelancer will charge you $300–$500 for a quarter page ad, and as much as $1000 for a full page, but your ad will be much more attractive as a result of this outside assistance.

The last and most lavish route is to contract with an advertising agency to design, produce, and place your advertisements. The smaller agencies may be willing to take on your business and for a typical advertisement will charge you several thousand dollars plus any production costs (such as photographer's fees). They also get a 15% "agency discount" from the magazine on the placement, which they collect by billing you for the full amount and pocketing the difference.

The larger ad agencies won't even talk to you, and here's why: They usually charge their clients nothing, or nearly nothing, to produce the advertisement; they make their income on the 15% or so "commission" on the ad placement. Consider the characteristic Borland four-color three-page foldout inside-front-cover advertisement. The production cost of such an ad pales into insignificance compared to the cost of running it in an array of magazines such as *BYTE*, *PC*, and *PC Tech Journal* (see Figure 1 again). Or for a really mind-boggling contrast with your own advertising budget, think back to the famous Apple "1984" ad that appeared on network television during the Super Bowl and kicked off their Macintosh campaign, and the revenues that must have been involved. You can then understand why the likes of Regis McKenna is not going to be too interested in helping you or me put together a spiffy quarter page black and white ad for a six-times insertion in *Programmer's Journal*.

One last point: Once you have ordered an ad insertion, it is totally your responsibility to get the ad copy to the magazine on time. If you reserve space and then fail to deliver the

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

materials, they are totally within their rights to charge you for the space anyway. Do yourself a favor and always ship your ad copy by Federal Express or some other overnight service. You will benefit from this in two ways: (1) you can prove when the ad arrived at its destination, thus eliminating potential arguments if you are cutting it close on a deadline, and (2) I really believe the ad is less likely to be misdirected or mislaid when it is delivered by courier rather than arriving in the magazine's post office box.

Handling ad responses: By the time your first advertisement appears, you should either be in a position to answer the phone during normal business hours or have some other pleasant human available to do it. An answering machine conveys an undesirable impression to prospective customers and should be used only as a last resort. In this month's installment, we will discuss information requests only; we will defer processing of phone orders to a future article. But bear in mind that if people go to the trouble of calling for information, a properly receptive attitude and the ability to supply the correct information on the spot creates a favorable impression and will often lead to an immediate sale rather than a delayed or lost sale.

However, most people who call or write in response to your advertisement won't make a snap decision to buy. You should formulate in advance a clear policy on what will be sent to people who respond to your advertisement by the various methods available to them. You will receive three basic types of inquiries for "further information":

- telephone calls
- personal letters
- "bingo cards"

Magazine bingo cards, where the reader can check off a number corresponding to an advertisement, lead to many casual inquiries. The reader is usually not an immediate potential customer because there is an implicit understanding between the reader, the magazine, and the advertiser that there is a significant delay (two or more weeks) in the response to such an inquiry. Most bingo card responses are from people who are simply curious or are filing away your sales literature for future reference.

Naturally, it is desirable to give the bingo card inquirers a good impression without spending too much money on them. We have found it expedient to create a small folded brochure that is sent to these readers along with a postcard that can be returned for more detailed information. People who return the second postcard, write a personal letter requesting information, or who make the effort to telephone are sent an 80-page catalog that describes each of our products in excruciating detail.

Again, a good way to get started is to look at what the competition is doing. Go through a recent magazine and send in your own bingo card for the companies with products similar to yours. Carefully review the material that they send you for both style and content, extracting the good concepts for cloning and taking note of the features you dislike.

Strive to make the best possible initial impression with your brochure. A crudely produced spec sheet printed on a dot matrix printer and xeroxed will alienate just about any potential customer. Your flyer should be typeset and printed on a heavy or glossy colored paper to make it easy to find on a cluttered desktop. What should a brochure or flyer contain? More specifics about the product of course; a detailed list of the advantages of your product over the competition's; and the exact information the customer needs to place an order: the phone number to call, the exact product name, part number if any, shipping costs and applicable taxes, and any special surcharges for air, overseas, or overnight shipping. State clearly which credit cards you accept and whether you will allow C.O.D. or purchase orders.

Assessing your ad results: You will be bitterly disappointed if you expect advertisements to yield instant results. Honest ad salesmen will admit that a software advertisement needs to appear three to six months in a row before most readers take it seriously. Computer magazine subscribers seem to have a remarkable mental filter for optical input that has not yet been discovered by neurologists; they simply don't register an advertisement until it has appeared in the same place for several months running.

Tracking the number of information requests that result from a particular ad, at least in a general way, is quite easy. Magazines summarize the bingo card responders on a weekly or bi-weekly basis and send you the names on pre-printed stickers—just count them up. Phone response to an advertisement is also easy to gauge—simply ask where the caller learned about the product. For the remaining category, the reader who wrote a personal note or letter requesting further information, you may find it helpful to put a distinct code in the address in each ad that will let you determine the source of an inquiry (such as "Dept. P" for the ad that runs in *Programmer's Journal*). Finally, if you include a user registration card in your product package, you can ask the user where he or she heard about your product (specific magazine, computer dealer, word of mouth, etc.) and thus relate actual sales to the original ad.

Paying the Piper: Assuming that you can eventually Get Credit, one thing magazines are earnest about is collecting on their advertising invoices. It is surprising how often these invoices are screwed up, though, so check them over painstakingly before writing out your check. Make sure the ad for which you are paying actually appeared (the magazine should send you a clipping with the bill), that it was reproduced properly, that you actually received any special placement that you may have been charged extra for, and that you are being billed at the appropriate rate for the size of advertisement and any frequency contract you may have signed.

Direct Mailing

If your product is directed to a well-defined subset of computer owners, direct mail advertising may be a relatively economical and effective alternative to magazine advertising. Several companies sell mailing lists sorted by occupation, type of computer owned, or geographic area. One such company is Irv Brechner's Computer Mailing Lists, Target Marketing, Inc., Box 5125, Ridgewood, NJ 07451 (phone 201-445-7196). The minimum purchase is 5000 names, and the cost per thousand ranges from $60 to $120. You can get a bulk mail permit from the post office that will allow you to mail two ounces for 12.5 cents if the pieces are properly sorted and bundled by ZIP code.

Again, it is crucial that your direct mail advertising look professional, supply the appropriate information about your product, and include all the necessary information for the customer to place an order. An order blank and possibly a "dis-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

count coupon" or "limited time special offer" price may increase the chances of an early response. References to favorable reviews in print or user testimonials are also helpful.

When assessing the results of your direct mail efforts, bear in mind that the industry considers a 2% conversion to sales on a direct mailing to be "normal."

Free Ink

Paradoxically, the methods of publicizing your program that cost the least are often the most effective.

New Product Announcements: Regular magazine sections that describe new software and hardware products, such as the "What's New" column in *BYTE*, typically draw their material directly from the manufacturer's press release—but since the material appears to be "editorial" rather than advertising, the reader gives it more credence. The major problem, from your point of view, is to make your product appear distinctive enough that your announcement will be one of the few selected for publication out of the hundreds received by the editors each month. Browsing through several months' worth of columns may give you a feel for the topics the editors are currently interested in (80386-related products, laser printers and desktop publishing, portable computers, and computer-assisted design programs are in favor at present).

Your press release should look professional and attractive. Print the original on a daisy-wheel or laser printer, or have it typeset, then have the local quick print shop reproduce the original directly onto your letterhead. Half-legible, typewritten, or dot-matrix printed announcements are diverted directly to the round file by most editors and columnists. An attractive color photograph will improve your chances of getting into the magazine; flashy screen images are preferred, but simpler color photographs of packaging, manual, and disks often get published too.

A mailing list of some 617 computer-oriented magazines and their editors can be purchased for $40 from Al Harberg at DP Directory, P. O. Box 562, Bloomfield, CT 06002. He also sells a mailing list of 185 IBM PC user groups for $20.

Reviews: A favorable review, from a source that the reader trusts, is equivalent to many thousands of dollars' worth of advertisements. Such a review, especially if it contains unambiguous information about hardware compatibility, positive remarks on the documentation, contact points for further information, and a kind word about your telephone support and responsiveness, will often prompt the reader to pick up the phone to order, send a check in the mail, or go down to the local software store and buy your product without further ado.

A product submitted for review should be completely finished, polished, debugged, and optimized, and the documentation should be complete. Just as a favorable review can make a huge difference in a program's success, a hostile review in a respected publication can be the kiss of death. The review that damns with faint praise, by approving everything about a program, say, except its speed, isn't much better. It's a good idea to put the software in the hands of ten or so friendly "beta testers" to thoroughly wring out problems in the program and holes in the documentation before sending it to any reviewers.

Magazine editors receive truckloads of software each month and can't possibly do anything useful with most of it. Therefore, don't just send your program blindly into the void addressed to "Editor, Programmer's Journal." Do a little research so that you can direct your review copy to the person best qualified to appreciate it. You are looking for writers who have demonstrated their interest and qualifications with (1) previous, fairly written reviews of your competitors' products, or (2) articles or columns about closely related subjects.

When you've identified such a person, contact him or her personally first, preferably by phone but if necessary by letter, to talk about your product and its advantages or novel features. Ask the potential reviewer specifically if he or she is interested in receiving a review or evaluation copy of the software at no charge or obligation and if so, where it should be shipped. This will minimize the chances that you will waste time and money by sending your program to someone who doesn't have the time or inclination to review it, or that the person will never receive it because it gets lost at the magazine's editorial offices.

Follow up the review copy by contacting the recipient (if he or she doesn't call you) to make sure it got there and that the recipient understands what it is, and answer any initial questions. Make this follow-up call very brief and objective. Don't ply the reviewer with questions about how he or she feels about the program so far, and don't take up too much of the reviewer's time. Many columnists write only as a sideline, wedged in among other, better-paying activities. Too much nagging or pressure on your part can be counterproductive.

However, be alert to possible calls for assistance or further information from reviewers. Try to insure that they will receive the most courteous and prompt responses possible. You can give a list of names to your secretary or answering service with instructions that calls from those people should be put through regardless of what else you may be doing.

Demo disks: One additional way of showing the public what your product can do is to create a restricted version of the program that can be distributed to ad responders or to targeted mailing lists. Put some real effort into it, though, so that the demo is utterly crashproof, flows smoothly, contains no typographical errors, and reflects the full power and polish of your product. A crudely produced demo is worse than no demo at all. Each demo disk will cost you about $1.00 including copying, mailer, and postage.

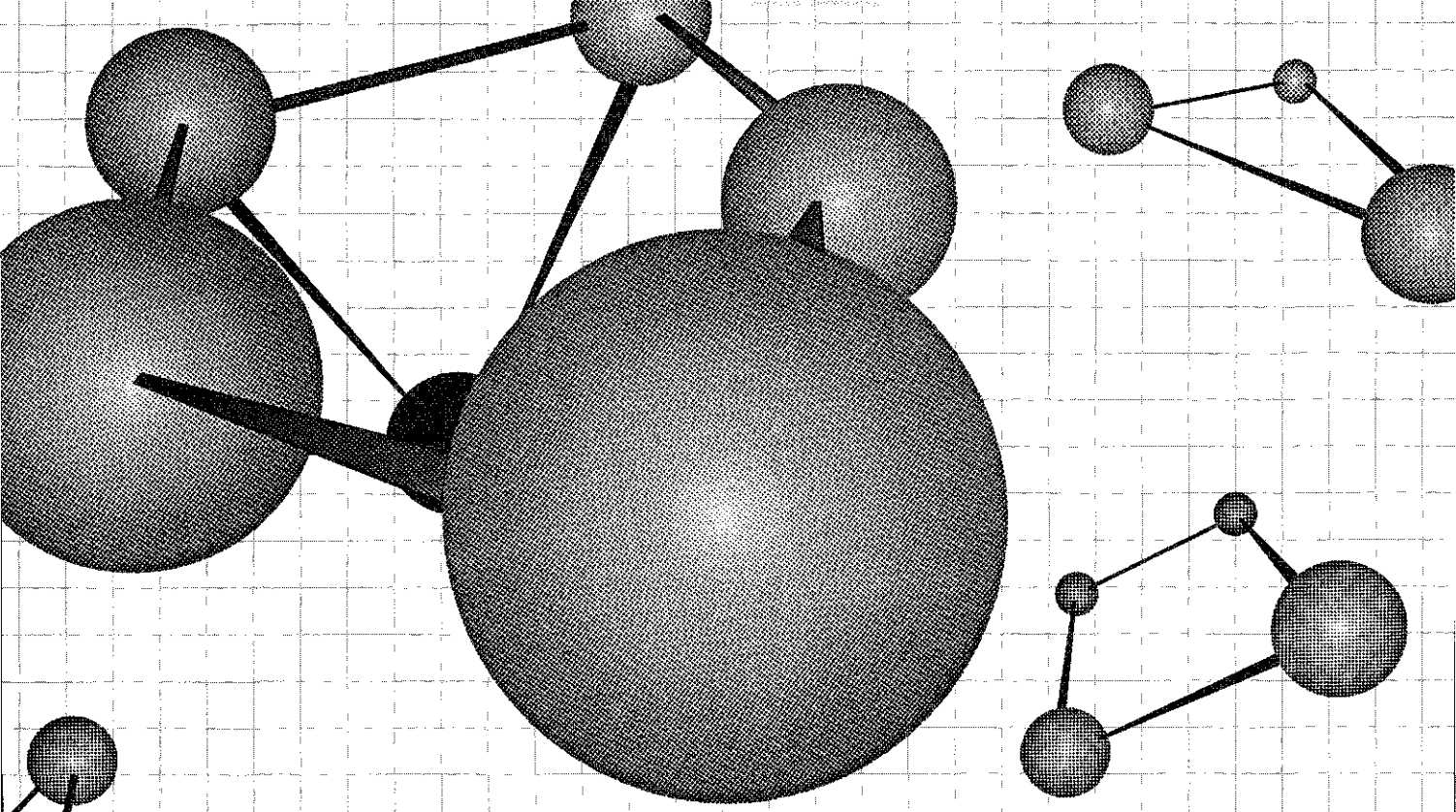
Another way to spread around a demo version of the program is to post it on popular local bulletin board systems, with a "Read Me" file that encourages people to propagate the program throughout the country by uploading it to other bulletin board systems that they frequent.

Future Attractions

In the next installment of this series, we'll discuss the really fun element of running your own software house: technical support. □

Ray Duncan is the president of Laboratory Microsystems and a regular columnist in Dr. Dobb's Journal. *His latest book is the best-selling* Advanced MS DOS, *published by Microsoft Press.*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Aztec C - the *best* C in the Micro-cosm

Why is Manx Aztec C the best C ?

Because Manx offers a combination of performance, features, and pricing that are unbeatable.

What kind of performance ? In the 2/87 issue of Computer Languages, our 3.2 release of Aztec C for PC-DOS turned out 109,489 iterations of the Dhrystone benchmark per minute to108,303 for Microsoft C. Our 4.1 code is even faster.

What kind of features ? Features like in-line assembly code, a professional make utility, library source, ROM support, debuggers, assemblers, and overlay linkers. Many of these features are unavailable or extremely high priced options in other systems. Microsoft, for instance, does not provide library source or in-line assembly.

What kind of pricing ? For $199, the Professional version of Aztec C gives you an Optimized C, a macro assembler, a librarian, overlay linker, extensive UNIX and MS-DOS specific library routines, a powerful debugger, sample programs, and over 500 pages of documentation. Manx Aztec C has a hard earned and well deserved reputation as a leading C software development system for PC-DOS and MS-DOS. At this price, it is also one of the best software values available.

Over 80 percent of our sales are repeat sales or are based on recommendations. That says it all. At Manx, we extend ourselves to the limit to provide the best software, the best service, and the best support possible. Call today, and join the over 50,000 users who know the Manx difference.

Manx also provides C systems for Macintosh, Amiga, CP/M-80, TRS-80, Apple //, 6502/65C02 ROM, 8080/Z80 ROM, 680x0 ROM, and 8086/80x86 ROM. **Call for prices and information.**

Aztec C86 4.1

PC-DOS • MS-DOS • ROM

Aztec C86-p Professional System $199

• optimized C with near, far, huge, small, and large memory • in-line assembler • in-line 8087/80287 • ANSI support • Fast Float (32 bit) • optimization options • Manx Aztec 8086/80x86 macro assembler • Aztec overlay linker (large/small model) • source level debugger • object librarian • MS-DOS 3.x file sharing & locking • comprehensive libraries of UNIX, MS-DOS, Screen, Graphics, and special run time routines.

Aztec C86-d Developer System.....$299

• includes all of Aztec C86-p • Unix utilities make, diff,grep • vi editor • 6+ memory models • Profiler.

Aztec C86-c Commercial System.....$499

• includes all of Aztec C86-d • Source for library routines • ROM Support • CP/M-86 cross dev. option • One year of updates.

How To Become An Aztec C User

To become an Aztec C user call **800-221-0440.** From NJ or international locations call **201-542-2121. Telex: 4995812** or **FAX: 201-542-8386. C.O.D., VISA, Master Card, American Express, wire (domestic and international), and terms are available.**

Aztec Systems bought directly from Manx have a 30 day satisfaction guarantee. Systems are upgradable by paying the difference in price plus $10. One and two day delivery is available for all domestic and most international destinations.

To order or for more information call today.

1-800-221-0440

In NJ or international call (201) 542-2121 • TELEX: 4995812

MANX

Manx Software Systems
1 Industrial Way, Eatontown, NJ 07724

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Unix Books, Part 2

by
Frank D. Greco

Frank Greco's survey of the Unix literature continues.

S.R. Bourne
The Unix System
Addison-Wesley

Steve R. Bourne is the author of the familiar Bourne shell so prevalent on Unix machines. It is this operating system/user mediator that really popularized the word "shell." Ken Thompson, the father of Unix, wrote the very first rudimentary command interface for Ma Bell's OS—in typical researcher fashion calling it the Thompson shell. Then parallel development at Bell Laboratories produced the programmable Bourne and Mashey shells (the latter written by John Mashey, who now works for MIPS). Shortly after some internal competition between the two interfaces, the Bourne shell became the standard line-oriented user connection to the Unix operating system. The most recent shell program from AT&T (as of System V Release 2) is the Korn shell, named after AT&T Bell Laboratories technician David Korn. The Korn shell is just a modernization (and a superset) of the Bourne shell and merely adds some bells and whistles.

In *The Unix System*, Bourne, who was employed at Bell Laboratories at the beginning of the Unix saga, gives an excellent succinct history of the Unix operating system. In his brief recap of the early days, he gives credit to David McIlroy, an unsung hero at the Labs who generalized the popular and useful notion of pipes. Though it's rarely mentioned, it was McIlroy who insisted on the symmetry of pipes from both a syntactic and a semantic point of view.

Bourne also notes that Brian Kernighan coined the term "Unix" in 1970 though he doesn't go into the etymology of the word. Is "Unix" an acronym or not? Although most people outside Bell Labs believe that "Unix" comes from something like "unified Multics" (Unix's forefather) or a "unified software development environment," the smart money has been placed on an off-color derivation. You see, Multics was an extremely large and bulky beast; most of Multics is written in feature-laden PL/I. Unix, thus, is a castrated or hacked down version of Multics.

In the chapter on editing files, Bourne gives only a cursory (pun intended) account of the now standard Unix screen editor "vi." Only eight pages are devoted to one of the most used first generation tools on a Unix system.

And keeping in mind that Bourne wrote the shell he describes, his shell chapter seems surprisingly "thin" (about 23 pages). Examples and illustrations of the theory would clarify points to interested students. However, most system programmers should find the chapter useful.

The chapter on the C language is concise, too, and for the most part accurate. Bourne's early Unix roots show up in the first substantial C program offered, an octal dump program. An octal dump program is not exactly useful these days. Bourne starts the beginning C programmer on an inefficient note besides, by using the bulky printf( ) output function to output a solitary space character. It would be more pedagogical to demonstrate some good, efficient programming skills by using the faster macro putchar( ). Bourne correctly describes the often misrepresented concept of precedence vs. order of evaluation: "The order of evaluation of expression operands and function arguments is not defined by C" (page 85). He points out that a compiler can arbitrarily rearrange an expression that makes use of the commutative operators: multiplication (*,/), addition (+,-), bit AND (&), bit OR (|), and bit EXCLUSIVE OR (◊); that is, "'a + b + c' may be evaluated as '(a + b) + c' or as 'a + (b + c).'"

I found a surprising mistake in the discussion of nested-if statements in this chapter. Bourne says (page 87), "the statement

```
if ( expression-1 )
     if ( expression-2 )
          statement-1
     else
          statement-2
```

is potentially ambiguous since the else part could be associated with either the first or second if. C chooses to associate the (dangling -FG) else with the first if." That's incorrect. C, like Pascal, associates the ambiguous else with the "most recently" nested if, that is, in the example above, the if with expression-2. A quick test indicates this quite clearly:

```
#include <stdio.h>

main(argc, argv)
char **argv;
{
     int i = atoi(argv[1]);
     int j = atoi(argv[2]);

     if ( i > j )
          if ( i == 5 )
               puts("arg1 is equal to 5");
          else
               puts("arg1 is NOT equal to 5");
}
```

Running this quick-and-dirty program (prog) demonstrates my point:

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
$ prog 5 1
arg1 is equal to 5

$ prog 6 1
arg1 is NOT equal to 5

$ prog 1 5                    # Note: no output!
$ _
```

The chapter on "Unix System Programming" touches on a lot of valuable concepts, but usually in a sketchy or superficial way. For example, Bourne insufficiently explains the concept of "semaphore files" or "lock files." In this common Unix practice a tiny file is used to synchronize multi-user (or multitasking) use of a particular resource such as a printer or modem. Bourne merely says (page 125), "A lock file is the exclusive possession of its owner and may be used to ensure exclusive access to a resource. . . .Creating the [lock] file so that it cannot be read ensures that another attempt to open it for reading will fail." This is confusing and slightly misleading. Consider Marc Rochkind's clearer explanation in his *Advanced Unix Programming* book: "When creat (the Unix file creation system call -FG) makes a new file, it is opened for writing even if the permission argument doesn't allow writing. When creat truncates an existing file, however, it fails if there is no write permission. This allows a file to be used as a semaphore, since such a creat succeeds if and only if the file doesn't exist."

In addition, Bourne spends only one page on the use of pipes. A pipe is a method of sending data between two related processes, another manifestation of the redirection concept. This fundamental concept did not even warrant a C language example. As a Unix system programmer, I was surprised to read (page 133), "The notation provided by the shell is adequate for most needs. . ." Most of you know the elegant shell syntax for creating a pipe—for example,

```
%  who | grep 'stu.*' | sort
```

This example shows a pipeline that will display a sorted list of currently logged-in users whose logins start with the string "stu" followed by anything else.

But how would one construct a pipe from a C program to initiate I/O to or from a Unix process? Using the standard Unix cursor control package "curses," you could create a simple windowing environment by reading the output of a Unix process (initiated by a user typing, for example, "ls -l") and possibly "clipping" the elongated output to fit in a 24-column scrollable window in the upper right hand corner of your terminal. Perhaps you want a better interface to an existing Unix utility: You could front-end the Unix utility with your new interface process. You can do both of these once you understand what a pipe really is. Unfortunately, Bourne gives this vital topic short shrift.

The "Document Preparation" chapter explains the old-fashioned Unix tools "nroff" and "troff," both descended from the "roff" text formatter available on the CTSS (Cambridge Time Sharing System) in 1965. Both nroff and troff are awkward, tedious tools, but Bourne at least makes them reasonably understandable. Of the two, troff (typesetting roff) will make a comeback as more and more laser printers conform to this older standard. Adobe Systems even has a tool that converts troff output to PostScript output so that you can use, for example, the powerful Apple LaserWriter on your Unix system. But with luck, newer Unix programmers will never have to use this dated assembly language for printers as more and more programs become, thankfully, fully PostScript compatible.

I found the chapter on "Data Manipulation Tools" to be the most interesting in Bourne's book. Here he discusses tools such as "uniq," "tr," "tail," "sort," "sed," "join," "grep," "diff," "comm," "cmp," "lex," "awk," "make," and "cron" and offers some clever shell scripts to develop a simple telephone directory, a C language cross-referencer, and a small database management system. The discussion of the DBMS is particularly interesting as it is designed to be self-maintaining. Here Bourne shines by combining most of the topics in the book (excepting system programming) to reinforce some potent Unix concepts. He combines several meaty shell scripts, the cron program scheduler, awk scripts, nroff, and C code to form the basis of a completely self-running DBMS *without* using a commercial database system. Of course a commercial database system will probably run faster, but we don't have to pay a dime for additional software; all the tools come with the Unix system.

After this chapter, I was eager for more on complete systems building with Unix. Unfortunately, the last 100 pages of the book are given over to appendices, that is, filler. Had Bourne written a sequel to the excellent chapter on data manipulation tools, he would have a winner here.

Meets Intended Audience	7
Title Matches Contents	9
Accurate	9
Thorough	5
Use of Illustrations	1

Strong Points	*Chapter 8 on complete systems building with Unix*
Weak Points	*The C programmer will feel cheated, Too many appendices*
Comments	*Bourne fails to reinforce topics until the very end, when he overcompensates*

McGilton/Morgan
Introducing the Unix System
McGraw-Hill

This "little black book" contains a monstrous 540 pages filled with a wealth of information for the Unix "user." The introduction describes the book as an introductory guide for users new to Unix. McGilton and Morgan don't explicitly define a "user," but from the looks of it, the "user" they address is, for the most part, the application programmer.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

McGilton and Morgan do a good job of grouping the usually confusing array of Unix topics into 14 compact chapters. The chapters range from "Getting Started" to "Unix System Administration." The ten pages of fine detail in the table of contents shows the authors' attention to the neophyte's point of view.

The first chapter is a concise, non-historical, and honest description of the Unix operating system for the beginner. I say "honest" because the authors correctly attribute the transportability of Unix to C: ". . .the Unix system is highly portable, because it is written in the C programming language." Most of the time, only a half-truth appears in the industry press—especially when an AT&T representative is quoted. Unix owes its reasonable degree of portability to the success of the C language. Let's face it, transportability was a key motive in the invention of higher level languages. Writing an OS in a HLL has its roots in the ill-fated Multics project, circa 1966; the bulky Multics OS was written mostly in, believe it or not, the IBM attempt at computer esperanto PL/I.

Towards the end of this introductory chapter, the authors inadvertently date themselves: "Because the Unix system is not the product of a computer manufacturer. . .there is no. . .wish to 'lock in' the user community. . . ." Supposedly, AT&T became a commercial computer manufacturer in recent years with the 3B2/5/20 "family" of processors. Then again, perhaps the authors remain correct. . . .

In discussing the terminal control command "stty," McGilton and Morgan introduce a little confusion. If the character-erase character is not the backspace or control-h character (the normal default) and you would like it to be the backspace, they recommend (page 19),

```
$ stty erase '^H'
$
```

and add, "Note that there are single quote signs around the argument. This is one of the places where you must use single quotes, and double quotes will not do. . . .the details differ from one system to another." Huh? Double quotes will do just fine here. McGilton and Morgan are a little confused about the quoting mechanisms of the shell interpreter. What happens is this: When you are typing things on the command line, you are talking to the shell, not to Unix; that is, any character that is special to the shell (a metacharacter), and there are loads of 'em, will be interpreted by the shell and will *not* be passed on to any Unix process. So if you want the stty utility to receive the caret character, which is a shell special character, you must protect it from being interpreted by the shell with single or double quotes. The difference between single and double quotes is this: Single quotes protect *everything* from the shell interpreter; double quotes protect everything except shell variables and command output. (Just for your information—the caret in the authors' example above is an outdated Bourne shell metacharacter used as a synonym for the shell pipe [|].)

For example, suppose we create four tiny files:

```
$ echo hello  >file1
$ echo there  >file2
$ echo my     >file3
$ echo friend >file4
```

And we have two shell scripts:

```
$ cat script1   # Display shell script number one
                    # Using double quotes
for i in file?
do
     echo " *** Displaying: $i ***
          'cat $i'
          "
done


$ cat script2     # Display shell script number two
                       # Using single quotes
for i in file?
do
     echo ' *** Displaying: $i ***
          'cat $i'
          '
done
```

Running script1 produces this output:

```
*** Displaying: file1
    hello
*** Displaying: file2
    there
*** Displaying: file3
    my
*** Displaying: file4
    friend
```

Running script2 produces this output:

```
*** Displaying: $i
    'cat $i'
*** Displaying: $i
    'cat $i'
*** Displaying: $i
    'cat $i'
*** Displaying: $i
    'cat $i'
```

Notice the difference. In script1, the shell variable $i was expanded to be a different filename that matched the criterion (file?) on each iteration of the for loop. In script2, the $i was taken literally as the two characters '$' and 'i.' Additionally, in the command output ('cat $i'), script1 expanded $i to the filename and presented that to a cat process whose output was subsequently presented to the echo process. Since script2 used the stronger single quotes, the command output 'cat $i' was not executed at all. It was taken literally. Therefore, script2 merely has an echo process that has a command line that contains no truly special metacharacters and spans several physical lines.

The work that the powerful and flexible Unix shell performs is certainly different from the awkward and inconsistent MS DOS command interpreter (command.com), where everything typed on the command line is presented to each process. Each MS DOS program must take it upon itself to expand filename wild card characters. In addition, each MS DOS compiler must adjust its own startup routines to accept multiply worded command arguments.

The "Programming the Unix Shell" chapter is really an overview of Unix shell programmability—not meaty enough for

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

my tastes. The power of the shell interface, either from the command line or in a script, cannot be underestimated. This is where the vast majority of Unix users utilize the power of the operating system. Hordes of Bell System employees who are everyday Unix users never see one line of C code; they do all their work from the shell level ("they never C below the shell"). Just as many PC users who regularly see only the Lotus interface consider themselves "experienced PC users," many Unix shell users consider themselves experienced "Unix users."

McGilton and Morgan start the shell programming chapter off in fine fashion by justifying the existence of shell scripts to the newcomer. The idea of creating your own "bin" directory with your favorite self-written shell utilities seems to explain itself in the authors' flowing description. Everything goes smoothly until M&M wind up their discussion on shell variables, which are merely macros. They say (page 425), "If you set up a shell variable as an abbreviation of a command line, the command must not contain pipe symbols (|), redirection (< or >), or the background processing symbol (&)." Why not? Of course it can! If you have a shell variable that has any of these symbols and you would like to execute it, you must place the shell built-in command "eval" in front of it to re-evaluate the command line for possible process spawning (redirection and background tasking involve spawning child processes under Unix). Since the shell normally evaluates (or parses) the command line only once, you must instruct the shell to re-evaluate the line since the first evaluation expands only the shell macro itself.

For example (assume the previous filenames: file1, file2, file3 and file4),

```
$ m='ls -l file? | wc -l'      # Assign string to m
```

This string when presented to the shell interpreter properly, will determine how many filenames match the file? criterion:

```
$ echo "$m"
ls -l file? | wc -l   # Yup, it's there all right

$ $m                  # Note the errors here
|: not found
wc: not found
-l: not found
file1
file2
file3
file4
$ _
```

When the $m was presented to the shell by itself, it was expanded to its literal representation. However, once that expansion (plus filename metacharacter expansion) is done, the shell cannot create a child process without re-evaluation. This is necessary because the shell must spawn a child process to execute the pipe. But if we do the following:

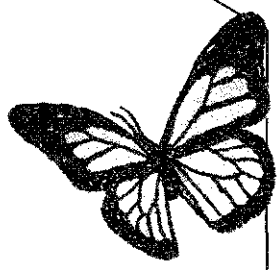
```
$ eval $m
      4
```

we force the shell to look at things again to consider a possible spawning of a child (whenever command redirection occurs) by prefacing the shell variable with the shell built-in "eval."

McGilton and Morgan's "Unix System Management Guide," the system administration unit, is loaded with valuable general information on how to care for a Unix system. Once you read it, you will understand the importance of looking after your

Chalcedony PROLOG

Not Copy Protected



A REAL Clocksin & Mellish Prolog for BOTH major microcomputing operating systems — with full cross-compatibility. $99.95

Complete with the predicates necessary for POWER AI programming:
op () name () functor () clause () =..("Univ")
...And no constraining data typing.

- Floating point
- Step-by-step tutorial
- Math functions
- Integrated editor

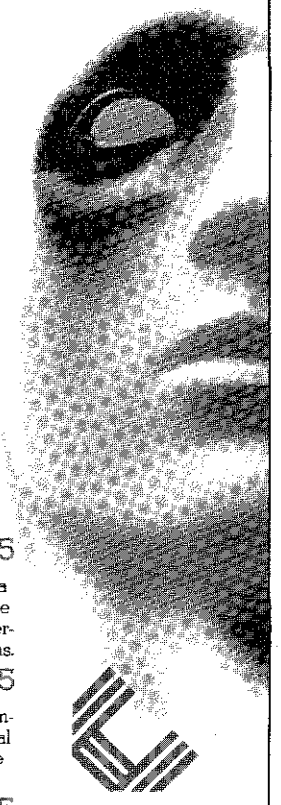
PROLOG/i

Extensible overlay library, 8087 support, large memory model (up to 640K)

PROLOG/m

Complete Macintosh environment with extensive pull-down menus and dialogue boxes.

No Risk Offer: Examine the PROLOG/i or PROLOG/m documentation at our risk for 30 days. If not completely satisfied, return with disk still sealed for refund.



APPLICATIONS — *Complete with SOURCE CODE*

NFL X-pert $49.95
A true interactive expert system written by a professional knowledge engineer. A valuable learning tool for any Prolog programmer interested in using Prolog to develop expert systems.

PROLOG ToolBox $29.95
More than 50 subroutines that speed and compress list handling, searches, sorts, and reversal algorithms. An inside look at the tricks of the professional Prolog programmer.

PROLOG ToyBox $29.95
Written by an academician to help his students understand Prolog, this collection of puzzles and mind-teasers will illustrate how the Prolog programmer creates programs that find the best solution to the problem. Turn your computer into a super reasoning machine!

CHALCEDONY SOFTWARE, INC.
5580 LA JOLLA BLVD.
SUITE 126
LA JOLLA, CA
92037
(619) 483-8513

System Requirements:
Minimum 256K RAM (320K recommended) PC DOS/MS-DOS ANSI Standard Support
Minimum 512K Macintosh Macintosh-plus and HFS Compatible

SAVE 10% when you buy either PROLOG/i or PROLOG/m and all 3 applications.

PHONE ORDERS: 1-800-621-0852 EXT 468

☐ PAYMENT ENCLOSED $ _____
CA residents add 6% sales tax
☐ CHARGE MY: ☐ MasterCard ☐ Visa
Card No. _____ Exp. Date _____
Signature _____
Mr./Mrs./Ms. _____ (please print full name)
Address _____
City/State/Zip _____

PROLOG/i	$99.95
PROLOG/m	99.95
Check:	
MS-DOS ☐	Mac ☐
NFL X-pert	49.95
TOOLBOX	29.95
TOYBOX	29.95
Complete Pack	188.82

SHIPPING:
$ 5.00 U.S.
7.50 Canada
10.00 Carribean, Hawaii Air
20.00 Overseas Air

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Unix resources. Be aware that this book was written *before* the general availability of Unix System V. There are substantial differences in the way one handles the administration of a System V Unix box. Thankfully, AT&T has made Unix System V administration reasonably easier to do and to understand than in previous versions.

One small issue in this chapter: The authors say, "A file in the Unix system is described by an object called an 'i-node'. We think that the name means 'interior node', since. . .Unix file-system is. . .a directed graph. . . ." To clarify the matter: The term "i-node" comes from "index node." Over the years the word "index" (or the embedded hyphen) was dropped in favor of a single lowercase word (traditional Unix style, of course).

Although the authors used a CP/M-based word processing system (SCRIBE/Scribble, Proofrdr and Grammatik) to produce their manuscript—not a Unix system—I still recommend this book to those readers who don't mind a lack of illustrations, a dry writing style, and a lot of general user-level Unix information.

Meets Intended Audience	9
Title Matches Contents	9
Accurate	9
Thorough	6
Use of Illustrations	1

Strong Points	*General Unix administration information, Good overview of shell programming*
Weak Points	*Lack of illustrations*
Comments	*Loads of Unix information, but hard to read*

Banahan/Rutter
The Unix Operating System Book
J. Wiley

Mike Banahan and Andy Rutter of the University of Bradford in Great Britain have written a dated book on the Version 7 flavor of Unix. Don't expect this book to help you out with the newer System V variants because Version 7 Unix is from the 1978 era and is more akin to the Berkeley varieties. I'm not quite sure which audience these British gentlemen were targeting when they put their quills to work. Their preface states that they write for the beginner and expert alike. I think they missed both markets with this mediocre work.

The *Unix Operating System Book* offers the obligatory explanations of "ls," "cat," "wc," and "who" commands; the usual few pages on shell programming; a gloss-over of the concept of software tools; and droll British humor: "Arguing about the 'best' language is about as useful as trying to decide. . .what makes a pretty sheep."

The chapter on "Process Environment" is well done but too spare. Banaham and Rutter undertake to explain the concepts of system calls such as fork( ), exec( ), exit( ), wait( ), access( ), signal( ), read( ), write( ), and the controversial setuid( ). They provide examples to flesh out their austere explanations, but thirteen pages really isn't enough.

An odd statement appears on page 121: ". . .any C program that uses routines from the SIO [Standard I/O, the authors' acronym -FG] must contain the line

```
#include <stdio.h>
```

near the beginning." That's not correct. If your program uses standard I/O library routines (or any library routines for that matter), it's the job of the linking program to find them and stitch them to your (user) routines. Kernighan & Ritchie's ubiquitous start-up program hello.c, which uses the stdio function printf( ), works under Unix without including a header file.

The last chapter on "Maintenance" gives the impression that the Unix filesystem is very delicate and easy to break. Here is where the version of Unix the authors are treating makes itself obvious. Thankfully, they alleviate some confusion by pointing out that they are describing Unix Version 7 on a DEC PDP11. The Version 7 filesystem was, honestly, very breakable. I did it quite a lot myself when I was learning Unix; luckily I was on a single-user 68000 Wicat machine, so I still have friends. The current versions of Unix have more robust filesystems, so it is harder to crash a Unix machine. Most of the administrative utilities mentioned in this chapter are outdated and have been replaced with more usable tools in the latest System V releases. I would read this chapter anyway because it gives you a feel for what a Unix administrator should do.

The *Unix Operating System Book* is a mediocre book sorely lacking in illustrations. It scratches its hooves but never charges.

Meets Intended Audience	3
Title Matches Contents	9
Accurate	7
Thorough	4
Use of Illustrations	1

Strong Points	*Unix process management explanation, System administration chapter*
Weak Points	*Minimal illustration*
Comments	*Very sketchy—not enough information to be useful, Discusses an older version of Unix—not really applicable to some System V material*

P.P. Silvester
The Unix System Guidebook
Springer-Verlag

P.P. Silvester justifies the subtitle of this book, "An Introductory Guide for Serious Users," in the preface: "People are mainly interested in putting Unix software to work effectively, hence need a good knowledge of its external characteristics but not of its internal structure." Well said. I wish Silvester had respected that perception and provided a good peripheral panorama of Unix. Instead, he mars his effort with the occasional unconventional explanation and tries to discuss Unix internals for the "serious user" with unsatisfactory results.

Silvester uses some non-standard terms when he discusses the Unix concept of "special," or device, files. Unix logically considers everything to be a file. Printers, modems, raw disks, tape units, even memory, are all considered files. This allows consistent high level manipulation of I/O and facilitates operations such as file redirection. The kernel recognizes these devices by having a bit set in the file's inode (recall that an inode is the heart of a Unix file). Also in the inode are disk addresses of the blocks that actually contain the data. Although a device file has an inode allocated for it, it obviously does not have data. So here is where the device type is maintained. The first byte represents the device type (a modem, for example) and is called the "major number." The second byte, called the "minor number," represents which device of that type (for

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

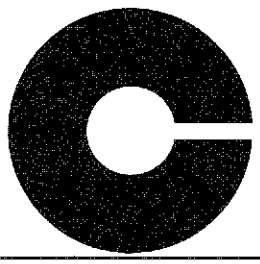
example, the second modem). When a user accesses that special file, the kernel knows which device driver to engage based on the major number and what parameters to pass to this device driver based on the minor number. When Silvester discusses special files, he unfortunately calls these numbers "device types" and "subdevice types." This nomenclature is certainly not what the rest of the Unix community uses.

On another topic Silvester says (page 79), "Disk files under the Unix system are physically organized into packets or blocks of 512 blocks." This is true under most Unixes, but not all. AT&T System V has a logical blocksize of 1K but still has a physical blocksize of 512 bytes. BSD 4.2 has varying blocksizes of 4K and 8K. In addition, because the blocksize is large under the Berkeley variant, the Berkeley gurus have devised a way of storing several small files in one block using something called "fragments." A block can be divided into 2, 4, 8, or 16 fragments that each can hold a tiny file. The operating system relates a small file to its fragments via a disk block bit-map.

In the section on "Input and Output" operations, Silvester confuses the serious user: "Certain file descriptors are allocated by default under the Unix operating system . . . File 0 . . . the terminal keyboard . . . File 1 . . . the terminal screen . . . File 2 . . . error messages" is certainly an external way of looking at Unix; however, it is misleading. Unix does not really offer default connections for standard input, standard output, and standard error. It is the shell program (actually shell process) that gives the Unix user these three defaults. When you login and start up the shell, it opens the terminal "file" three times: standard input for reading, standard output for writing, and standard error for writing, respectively. Each open seizes an entry in the (per-process) file descriptor table. From that point on, all other processes that you start up are child processes of the shell or perhaps child processes of child processes. When a process is created in Unix, it inherits the open files of its parent. So unless you alter ("redirect") the file descriptor table, all child processes of your login shell will have the first three file descriptors already opened and connected to the terminal.

In Chapter 5, "The System Kernel," Silvester says that a Unix file is simply a string of bytes, terminated by an end-of-file mark (control-D). While this is certainly logically correct from an application programmer's point of view, it is not really the truth. The essence of an opened Unix file is contained in two kernel data structures, called the system open file table and the in-core inode table. These data structures contain, among other important items, a current I/O "pointer," which tracks the position of the next byte to be read or written, and the filesize in bytes. Whenever a read is done from a file, the kernel compares this "pointer" to the total byte size. When the I/O "pointer" exceeds the filesize, end-of-file is asserted. There is no physical end-of-file mark per se.

Another high level but misleading statement is made in regard to the "at" utility. The at utility allows a user to schedule an event to occur at a specified time, day, or date. It is available only on older versions of AT&T Unix (Version 7) and Berkeley



the dBx™ Translator

dBASE to C

- dBx produces quality C direct from dBASE II, III, or III+ programs.
- Move dBASE programs to UNIX or other machines.
- Improve program speed and reliability.
- Support multi-user/network applications.
- Includes full screen handler and allows a choice of C database managers.
- May be used to move programs, help dBASE programmers learn C easily, or as a daily tool.
- For MSDOS, PCDOS, UNIX, XENIX, Macintosh, AMIGA or any standard C system.
- Priced from $350; also available from distributors.

from Desktop Ai 1720 Post Road East, Westport, CT 06880
Telephone: 203-255-3400 • Telex: 6502972226MCI
MCIMAIL-DESKTOPAI

dBASE is a trademark of Ashton-Tate dBx is a trademark of Desktop Ai

Number 103 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

flavored Unixes. Silvester says, "The kernel periodically looks for pending at commands." More accurately, the Unix kernel runs the at utility as it does any other program; it has no notion of "pending at commands." There is a process called "cron" that is started (usually) when the machine is booted and never exits—a "daemon" in Unix parlance. Whenever cron's time slice comes up, it inspects a table for things that should be done. Sometimes this table is called atrun or crontab. If an event is due (or overdue), it is started up. The nucleus of the Unix operating system does not *know* about the at utility any more that it *knows* about ls, cat, grep or informix.

The explanation of Shell input handling is awkward and deceptive. Silvester's metaphor to elucidate the shell command line concept—'think of a pipe'—is clumsy. His statement that the "buffer area used for shell input is 256 characters in size" is partially correct. Silvester is referring to the 256-byte external buffer area that the user sees. The internal buffer that the shell uses is 5120 bytes. This can handle the expansion of many large shell variables or filename matching from metacharacter expansion. In addition, the erase and kill-line characters are handled by the tty device driver, not the shell. The allocation of a paltry four pages for shell programming concepts is worse than a simple oversight.

Another oddball statement that echoes some of my college professors concerns the use of HHLs: "Since C does nearly everything necessary for system programming, there is little need for elaborate macro assemblers under any operating system that supports C." That's a naive statement. Surely a *real* programmer would not make that claim. When I read that statement, I went to the front of the book to find a bio of the author. Sure enough, Mr. Silvester is employed by the EE department of McGill University in Canada. 'Nuff said.

Meets Intended Audience	3
Title Matches Contents	7
Accurate	4
Thorough	3
Use of Illustrations	0

Strong Points *Discussion of f77 and Ratfor in Unix environment*

Weak Points *Too sketchy, especially in the shell section*
Comments *Attempts to discuss Unix internals are faulty* □

Frank D. Greco is an independent consultant who specializes in highly usable computer systems.
Readers may contact Frank at the following address:

Frank D. Greco
P.O. Box 470
Kearny, NJ 07032-470

Introducing Periscope™ III

A new generation of debugging for the IBM PC, XT, AT and close compatibles

Now you can invest $995 and get the most powerful debugging tool available short of a $10,000 in-circuit emulator! The Periscope III board's hardware breakpoints and real-time trace buffer help you solve the really tough debugging problems. If you ever deal with errors in real-time systems, intermittent failures, interfacing with undocumented systems, or bottlenecks in your code, Periscope III may be just what you need!

Call TOLL-FREE 800/722-7006 for more information.

The PERISCOPE Company, Inc.
14 Bonnie Lane • Atlanta, GA 30328 • 404/256-3860

Number 6 on the Reader Service Card

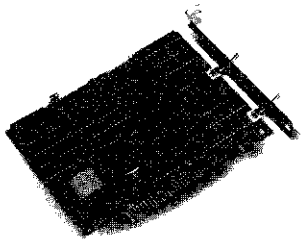
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MICROWAY ACCELERATES YOUR PC!

FastCACHE-286™

Runs your PC Faster than an AT!

Runs the 80286 at 9 or 12 MHz and the 80287 at 8, 9 or 12 MHz. Includes 8 kbytes of 55ns CACHE.



Compatible with Leading Edge Model D, Compaq, and Turbo motherboards. Includes 8088 Reboot Switch, DCache, Print Spooler and Diagnostics... **From $449**

LOTUS/INTEL EMS SPECIFICATION BOARDS

MegaPage™ The only EMS board which comes populated with two megabytes of cool-running, low power drain CMOS RAM installed. Includes RAM disk, print spooler, disk cache and EMS drivers. For the IBM PC, XT and compatibles...**$549**

MegaPage with ϕK..... **$149**

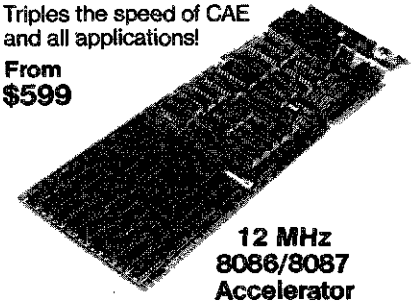
MegaPage AT/ECC™ EMS card for the PC AT and compatibles includes Error Correction Circuitry. With ECC, 11 RAM chips cover 256K so the user never encounters RAM errors. Sold populated with 1 megabyte CMOS . . . **$699** or with 3 megabytes CMOS cool running low power drain RAM . . . **$1295**. Optional serial/parallel daughterboard..... **$95**

INTEL, JRAM, or Maynard..... CALL

NUMBER SMASHER/ECM™

Triples the speed of CAE and all applications!

From $599



12 MHz 8086/8087 Accelerator Plus A Megabyte for DOS!

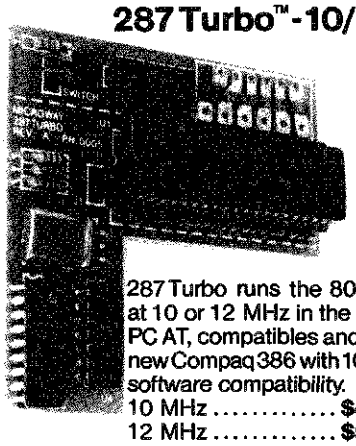
For the IBM PC, XT and compatibles

***PC Magazine* "Editor's Choice"**

8087 SOFTWARE

IBM BASIC COMPILER	$465
MICROSOFT QUICK BASIC	$79
87BASIC COMPILER PATCH	$150
87BASIC/INLINE	$200
IBM MACRO ASSEMBLER	$155
MS MACRO ASSEMBLER	$99
87MACRO/DEBUG	$199
MICROSOFT FORTRAN V4	$299
RM FORTRAN	$399
LAHEY FORTRAN F77L	$477
MS or LATTICE C	CALL
STSC APL★PLUS/PC	$450
STSC STATGRAPHICS	$675
SPSS/PC+	$695
87SFL Scientific Functions	$250
87FFT	$200
OBJ → ASM	$200
PHOENIX PRODUCTS	CALL

287 Turbo™-10/12



287 Turbo runs the 80287 at 10 or 12 MHz in the IBM PC AT, compatibles and the new Compaq 386 with 100% software compatibility.

10 MHz **$450**

12 MHz **$550**

***PC Magazine* "Editor's Choice"**

8087 UPGRADES

All MicroWay 8087s include a one year warranty, complete MicroWay Test Program and installation instructions.

8087 5 MHz..... **$114**
For the IBM PC, XT and compatibles

8087-2 8 MHz..... **$154**
For Wang, AT&T, DeskPro, NEC, Leading Edge

80287-3 5 MHz..... **$179**
For the IBM PC AT and 286 compatibles

80287-6 6 MHz..... **$229**
For 8 MHz AT and compatibles

80287-8 8 MHz..... **$259**
For the 8 MHz 80286 accelerator cards and Compaq 386

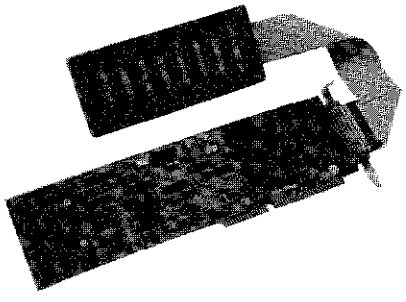
80287-10 10 MHz..... **$395**

PC-PAL™ Programmer..... **$395**

Call for great prices on V20, V30, 64K, 128K and 256K RAM

AT8™

Turns your AT into a high speed, multi-user Xenix business system!



8 port, intelligent serial controller with 3% response degradation. Includes 8 MHz 80186 with built in DMA **$1299**

MICROWAY SOFTWARE FOR LOTUS 1-2-3™

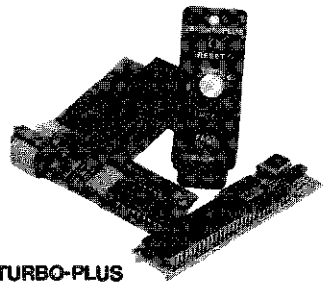
FASTBREAK™ employs the 8087 to increase the speed of Lotus 1-2-3™ Version 1A or 1A*. Users are reporting speed ups of between 3 and 36 to 1. When run with our NUMBER SMASHER accelerator card, recalculation speed ups of 10 to 30 are being reported..... **$79**

PowerDialer® Add-In for Lotus 1-2-3 Release 2. Automated telephone dialing from within 1-2-3. Adds least cost routing, automatic carrier selection and automated phone book worksheet. Builds customized dialing applications. Can be used with DesqView **$79**

HOTLINK™ adds easy linking of spreadsheets to Lotus 1-2-3 Version 1A... **$99**

287 TURBO-PLUS™ Speeds up your AT

Adjustable 80286 Clock 6-12 MHz
10 MHz 80287 Clock
Plus Full Hardware Reset..... **$149**
Optional 80286-10 ..... **$175**



287TURBO-PLUS
With 80287 10 MHz..... **$549**
With 80287 12 MHz..... **$629**

CALL (617) 746-7341 FOR OUR COMPLETE CATALOG

MicroWay P.O. Box 79 Kingston, Mass. 02364 USA (617) 746-7341

You Can Talk To Us!

MicroWay Europe
32 High Street
Kingston-Upon-Thames
Surrey England KT1 1HL
Telephone: 01-541-5466

Number 104 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Merger Mania

by
Paul Barkley

Companies are a lot like fish. They start out small, and they grow. Eventually, every fish either gets eaten by another fish, meets some other untimely death, or grows large enough to swim with impunity in the company of the large fish.

Back to business. I remember a touching series of ads that showed two kids opening a lemonade stand and then flashed to scenes of increasing scope: a small roadside fruit stand, a larger fruit and vegetable stand, a small grocery store, a larger grocery store, and finally a supermarket. Each new business growth brought a corresponding change in signage as well—the manifestation of growth and sophistication. What a country, huh?

Computer companies seem to follow similar growth patterns. Everyone knows about the growth of Microsoft, Ashton-Tate, and Lotus—the three giants of software publishing. The tales of Televideo, AST, and other successful hardware companies are heartening, too.

Unfortunately, there have been some spectacularly unhappy endings in all this fairytale growth (Osborne Computer comes to mind), so the prudent person will no doubt recognize that growth per se is not necessarily good. If you're bored some rainy day, have an MBA explain what happens to cash flow when business is booming and you're forced to rapidly expand to keep up with it. It takes only a few months of cancerous growth to sink even the healthiest company, and a quantitative analysis will demonstrate that clearly.

Doom and gloom aside, many consultants are interested in expanding their businesses. Although there are a number of smug, successful independent consultants who adamantly oppose hiring any employees, taking on work assignments that are not seductively interesting, or doing anything remotely distasteful (to them) to earn a buck, these people seem to be in the minority—they're folks with individualistic and perhaps isolationist mentalities that make them perfectly satisfied with their current work-styles.

Getting Rich

Most people, though, want to know how to get rich, preferably quickly, painlessly, and without risk. Other than selling gullible people books that purport to teach just that, there is obviously no easy way to get rich—otherwise, everyone would be rich, and by definition that's impossible.

The generally accepted path to wealth in consulting is to either branch out of consulting into a related field, such as software publishing, or to greatly increase the size of your consulting business. Keep in mind that no matter *how much* you charge per hour for your services, you can make only a finite amount of money because you can work only so many hours per week. While neurosurgeons and other highly paid professionals make good money, they don't make nearly as much as successful businessmen because they, too, are limited to the work that they can personally accomplish. When they get old and out of work, their incomes stop. Of course by then they should have sizable investment portfolios that allow them to live comfortably—but that's another story.

Branching out of consulting is attractive for the simple reason that anything other than what you're already doing always

PRODUCED 2007 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

looks good—"the grass is always greener on the other side" syndrome. When I was in retailing, all the retailers I knew thought that wholesalers had it made because they sold in large quantities with few support problems. Aggressive retailers always managed to start distributing some line of products on the side, usually disastrously. Similarly, the wholesalers thought the manufacturers lived on easy street because all they had to do was to make box loads of the same thing all day, instead of buying from a bunch of different people. Aggressive wholesalers soon had similarly disastrous minor manufacturing plants going.

So I would caution you about getting too excited about a related field, such as software publishing, unless you understand what's involved. I market a cosmetology school package that tracks student progress in a clock hour environment (attendance, task, test, class accomplishments), and selling it is seductive—every time another one goes out the door I feel like I'm getting rich.

On the other hand, all the small (one or two man) software publishers that I know consult on the side. They consider themselves publishers and not consultants, but consulting puts food on the table. If publishing is such a great field, I ask myself, why aren't these guys as rich as Mitch (Kapor)?

Growing from One to Two

Back to consulting: If you want to expand your consulting business, and I include custom software development here since that seems to be an integral part of the average consultant's repertoire, you need to make the transition from a one man (or one woman) firm to a multi-person firm. Aye, there's the rub.

How does one person take on the services of a second person? Well, you can either double your business overnight, or cut your share in half—take your pick. Cutting your share, considering the fact that you still have the same fixed overhead, or more, probably means that your net approaches or drops below zero for a while.

The only way you're going to do this successfully is to have a large contract lined up or to have saved a lot of money from your previous years in consulting. (Some people do manage to bank a substantial portion of their earnings, I'm told, but I've never seen this. All the people I know always manage to increase their spending to at least their current income level.)

Bidding on a contract that's larger than you, as a single person, can handle is also tricky because of several things. First, you almost have to lie about the size of your firm to be considered in the first place, and I absolutely do not recommend that approach. Simply skirting the question can't usually be sustained through the proposal or contract acceptance stage unless the client is not very smart, and you may not want a client like that anyway.

If you do manage to get the contract, though, you now have to scramble to find someone to help you with it, and this is a high pressure way to start. Your odds of satisfactorily completing the contract on time are not real hot—although it does happen, of course. You can sub-contract pieces, but generally a consultant can't afford another consultant's rates unless there's a considerable discrepancy in seniority and therefore in rates as well.

So, getting that first person on your staff is really tough. If you already had, say 5 people, adding a sixth would be relatively trivial. One more person is proportionally a small change in your current staff. You can bid a project that takes 6 instead of 5 people with relative impunity. Reaching a size (almost any size over one) allows you to grow incrementally, with each potential increment becoming smaller as your size grows. Decrements, too—that's why a large company can trim 4,000 people from its staff with relatively little disruption (except for scaring the hell out of the other 96,000).

So it seems that if you could grow just to four or five people, further growth would be easy. (There is obviously a whole new set of problems when you get to that size, such as how to grow to 20 people, but we won't worry about that right now.) Let's set aside the question of how to get there for a moment and talk about some other problems associated with being a one person firm.

Economies of Scale

I find my hours of marketing effort to be disproportionate to the amount of my billable time. In order to bill hours, you've got to keep attracting clients. Slightly over half of my revenue comes from clients for whom I've previously done work—not referrals, mind you, but follow-on work from the *same* clients. That tells me that if I can land a new client, I not only obtain the value of the current contract, say $4,000, but can count on, statistically, another $2,000 in business sometime in the future.

It also tells me that if I didn't get any new clients, my income would drop by half. Many of my clients are very small businesses who only have one task that I can perform for them, and when that's done, I've worked myself out of a job. Also, most of my small business clients are the most progressive in their fields, and they don't know anyone else who can use my services because they're so far ahead of all their peers. So, while they're willing to serve as references, they don't generate any referrals themselves. (This is not usually the case with consulting practices where the clients are much larger companies—company to company referrals are more prevalent.)

Consequently, marketing activities must go on. This means phone calls, letters, brochures, attending organization meetings of both peers and potential clients, advertising, and so forth. Quite an effort for one person, but ironically almost the same amount of marketing effort could serve a small group of people just as effectively. For example, meeting a potential client or planning and executing a direct mail campaign has nothing to do with the size of your firm. In the latter case, it's merely a matter of sending out 500 pieces, say, instead of 100 pieces—a trivial difference in time compared to the total time spent planning and executing the campaign.

Another benefit of having a larger firm is access to more experience—if you don't have the answer, perhaps someone else in your firm does. You can divide up projects so that each person works on an optimal piece, for example, according to his or her skills. Let the high-level specialist do the high-level stuff and the low-level people do their things as well.

You can share physical resources better, too. Everyone needs a conference room from time to time, but a one man firm doesn't need one very often. A single conference room can probably accommodate twenty people's needs. You can share hardware, too. How many laser printers do you need? How about EGA monitors or plotters or modems or scanners? Everyone has an occasional need for unusual (to them) hardware but can't justify buying it for his or her own use. Buying one laser printer for five people to share makes a lot more sense.

There's another advantage, and I hear it increasingly from consultants who are willing to be honest about their day to day

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

New Version 2.0

Complete C Programs in Half the Time, with *Instant-C*™

You can create programs much faster with Instant-C than with conventional programming tools. How? Because Instant-C is a high-performance interpreter, there are **no compile or link delays.** Change your program, then test it immediately. No matter how large your program, the turnaround time is just seconds.

"Instant-C means instant gratification."—*PC Magazine,* **Editor's Choice** for best C interpreter.

Powerful **source-level debugging** saves your time. Conditional breakpoints, single-stepping by statement, source code backtraces, data monitoring, and many other debugging features make it easy to wipe out bugs quickly. Direct execution of any statement or function makes testing a breeze.

"The resulting debugging and testing capabilities are fantastic and the detailed trace/debug/display commands make it easy."—*The C Journal*

Instant-C checks pointer references for reasonableness, and checks that array indexes are within declared bounds. This **run-time checking** stops your program as soon as errors occur, for easiest debugging.

Not only does Instant-C help you quickly change, test, check and debug your code, but it runs your program **fast enough for real-time** applications.

"It is much faster than any of the other products mentioned and was the only one able to complete the standard SIEVE in a reasonable time. Clearly, this high speed allows much more complex problems to be attacked with Instant-C than with any of the other products discussed."—*Computer Language*

Immediate feedback and precise diagnostics make Instant-C great for learning C. Full K&R and the ability to **link compiled object code and libraries** (Lattice and Microsoft) makes Instant-C compatible with your existing programs.

Instant-C makes all parts of the programming task as fast as possible.

"Clearly, Instant-C is the performance champion."—*PC Tech Journal*

Version 2 works with MS-DOS and PC-DOS, and has a full 31 day **money back guarantee.** Instant-C is only $495. Order today! Call or write for full information.

Rational Systems, Inc.
P.O. Box 480
Natick, MA 01760
(617) 653-6194

Number **20** on the Reader Service Card

activities—they need someone to talk to. Sometimes it's to "bounce ideas off" someone, and sometimes it's simply to talk about problems with clients—just letting off steam, not finding an obscure technological solution.

Many psychologists and psychiatrists operate in group practices. Although they have certain economies of scale, too, one important reason is to keep or re-establish their personal perspectives. After a day of listening to nothing but problems, many groups gather for a scheduled daily wrap/rap session. The purpose? To get their heads screwed back on straight.

How do *you* feel after a long day of listening to computer problems? Do you feel exhilarated because everyone's system is in deep trouble and you're going to make a lot of money fixing things? If so, good for you—you should teach everybody else how to acquire your myopic mindset—you'll make a fortune. Most consultants with any sensitivity often feel that the entire world of computers and data processing is going to hell in a hand basket because all reports are negative. All consultants hear is one problem after another, from disk crashes to blown indexes to tales of old bottleneck systems that are ruining productivity. When was the last time one of your clients called to tell you how wonderful everything is, and how there hasn't been a problem in months and months, and for you to just have a nice day?

I'm always cheered up by chatting with other consultants about their crazed clients and the harm they've caused their systems. Why? Because somebody else has to deal with the problem, that's why! I can tell them about the ridiculous situations that I have to deal with, and I get sympathetic nods, too. Instant group therapy for the battered consultant.

How Did Other Firms Grow?

Okay. Now that we've established the many advantages to becoming a larger firm, how do we get there from our current one person outfit? The obvious solution is to look for models. How did other consulting firms get to be the size that they are?

There are a lot of one person firms, but there aren't many larger firms, for a variety of reasons. Interestingly enough, many consulting firms grow and stop doing very much consulting. They become vendors of semi-customized systems, or outright software publishers, or body-shops.

One consultant I talked to suggested looking at TRW and Booz-Allen as corporate role models—not quite the scale I had in mind. In fact, of the few larger computer consulting firms that I know, virtually all grew from one person companies because of some completely fortuitous piece of business—a unique situation that serves us not at all as a model for our own behavior.

Looking further afield, though, we do find a lot of role models in other professions, particularly in accounting and law. In those fields, the norm is to form partnerships and grow from there. The benefits are *exactly* the same as for a larger computer consulting firm: economies of scale.

Note that when I said that these models are organized as partnerships, I did not say that their *legal* structure is a partnership. Any beginning business book will caution you to avoid setting up a legal partnership structure because each partner becomes responsible for the liabilities of the others, and in our litigious society that means you could get stuck for the disastrous effects of your partner's drunk driving.

Although the "feel" of the firm is a partnership, the actual legal structure is generally a corporation, or a special sub class known as a "Professional Corporation," called a "PC" (droll acronym, huh?). The partners have shares of stock and receive profits in proportion to their shares, as in any corporation.

When I started investigating how law firms structure their partnership agreements, I found that the structure varies tremendously from firm to firm, and I found that there are a number of pitfalls.

First, one partner has to be the manager of the business—nothing will get done if every decision is made by a committee. This person should probably do the hiring and firing as well. The managing partner should get input from the other partners, but should handle all of the routine matters that come up.

The first problem you run into is how to actually split the profits. If one partner puts up a lot more capital than another, the capital partner needs either some direct interest return based on that capital, or, more likely, a higher return by way of more shares of stock. Similarly, allowances should be made for partners who manage or have other duties in addition to directly bringing in revenue. In some firms, each partner receives a percentage of the fees that he or she directly bills and then shares in the profits of the firm overall. Thus, a hard working partner takes home more than a laid-back partner.

Some firms pay a percentage fee to the partner who brings in the business and a percentage to the partners who do the billable work. Since bringing in clients is a

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

crucial activity, this seems reasonable. Fees can range up to 25%, though, and I've heard of a number of dissolved partnerships where one partner became more a "salesman" and spent most of his time bringing in clients and not doing any work. This marketing partner enjoyed hanging out at the country club, taking prospective clients to lunch, and so forth.

This would have been okay except that the other partners, who were busting their tails to get the work done, came to resent what they perceived as a leisure lifestyle. In one case, the partnership dissolved because the marketing partner kept promising things that couldn't be delivered, taking his percentage and leaving the firm as a whole with unhappy clients and a mess to clean up. This is the classic schism between marketing and engineering. I suspect most of you readers are (rightfully) suspicious of marketing types, anyway.

Other problems involve, naturally, loss of control. Instead of being fiercely independent, now you have others to answer to, and this inevitably means that you don't always get your way. This may not be acceptable to some people.

I've heard of only one instance of a computer consulting partnership failing, but keep in mind that since unsuccessful firms disappear, they're soon forgotten. No doubt the percentage is fairly high. The fatal flaw in this particular company was a large increase in overhead. Three consultants joined their home-based practices and went out and rented office space and office furniture, got new business phone lines and stationery, and turned their three almost zero overhead businesses into one real cash-eater. A month's downturn in business sank one partner, who failed to make his contribution, and within three months the whole thing went down the tubes because the remaining two partners couldn't (or wouldn't) carry the whole thing. Their biggest problem was that no one already had an office, and all three were running very marginal businesses with little room for error. The smallest screw-up in the business plan (assuming that they had one) was deadly.

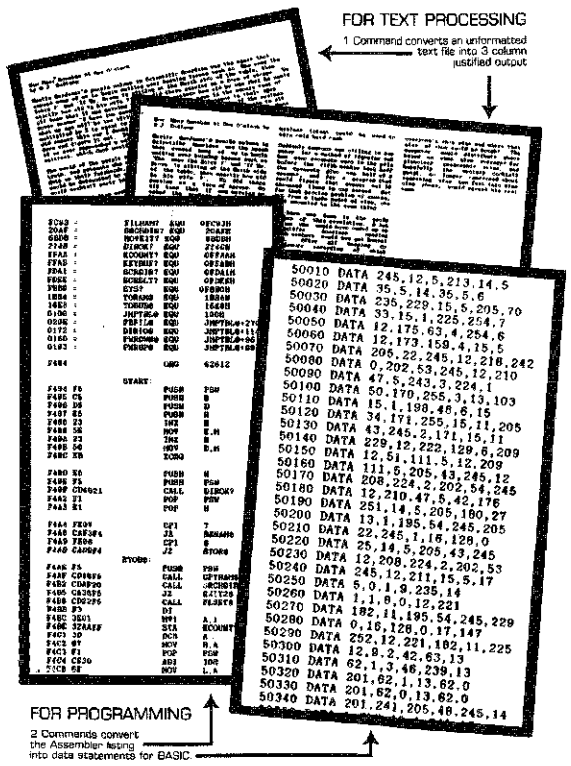
When I asked the senior partner of a successful twenty person (three partner) computer consulting firm how and why he made the first step, he told me that he'd really wanted the services of the two people who were to become his partners. He respected their individual talents, and their skills complemented his own. The only problem was that he couldn't possibly afford to hire them, so he made them partners instead. This way they could all stay poor together (while building up to fifteen or so employees). He said "stay poor together" not entirely tongue-in-cheek, so remember that growing may not make you rich right away.

I believe there is a threshold level below which any business will have substantial long term difficulty surviving, and for one person consulting firms, that threshold is somewhere above them. Although business might be good now, the smallest change in market conditions, technology, or your own health will cause dire consequences for your firm. While disability insurance might solve some of your problems, consider the alternative of merging with other like-minded consultants.

What Happens When You Don't Grow

Finally, let me note that patterns of merger and acquisition among computer firms (and non-computer firms) of all sorts suggest a dim future for very small businesses. Ashton-Tate and others are devouring firms at an alarming rate. Also, the single computer dealer is in deep trouble as the trend toward merger and acquisition heats up. In a few more years, most dealers will be parts of larger companies—independents are already growing scarce as many close their doors.

The situation is more easily seen in the case of independent grocery stores.


FOR TEXT PROCESSING
1 Command converts an unformatted text file into 3 column justified output
FOR PROGRAMMING
2 Commands convert the Assembler listing into data statements for BASIC

CUT HOURS OFF TEXT PROCESSING TIME WITH:

TEXT-PRO

Use Text-Pro to make reusable, canned multiple Search & Replaces to use on different files.

Create Batch files, reformat, select, merge, split, sort files, insert records, do Global Search & Replace without going into the editor.

Convert variable to fixed length records & vice versa.
Produce documentation with real impact.
Format listings.

Makes light work of Code & Data Conversions—handles Code conversion problems with a single command.
Ideal for porting Code.

Chain several of Text-Pro's 29 programs together to produce dramatic changes in Text files.

Use to speed programming, conversion, text editing.

In use in major Electronic Publishing firms & Software Houses.

For MSDOS & PCDOS machines.

King Computer Services, Inc., 1016 N. New Hampshire
Los Angeles, CA 90029 (213) 661-2063

Number **91** on the Reader Service Card

PRODUCED 2022 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Everyone knows what happened to the mom and pop stores when somebody got the bright idea to build a supermarket. Mom and Pop were forced out because they couldn't compete on either price or breadth of selection (think about those two factors in your own business). Similarly, in a few more years there probably won't be many, if any, single car dealerships around—they're all being acquired, and the manufacturers are implicitly supporting this trend as well. Most are not only multiple-site dealers, but they offer multiple competing car lines at each site as well. Price and selection. Economies of scale.

What happened to grocery stores and car dealers and computer dealers and software publishers is not necessarily what will happen to independent computer consultants, but the trend is certainly there. There will always be contract programmers and at least a few independent consultants, just as there are still some quaint little grocery stores here and there that won't go away, but the aggregate long-term picture is not good. So, even if you're not sure that all the benefits I've talked about will outweigh the disadvantages, consider your long-term survival as well, and remember what always happens—to little fish. □

If you have consulting firm merging or acquisition experiences you'd like to share, or have comments or questions, please send them to

Paul A. Barkley
The Barbican Company
6845 Elm Street
Penthouse One
McLean, Virginia 22101

#1 C interpreter

C-terp

The professional C development environment

Your C compiler creates great final code . . . but as a programming tool, it's too, t o o s l o w. With C-terp you can edit, debug, and run without the wait. Nothing, but nothing, is faster for developing professional C programs.

Choose the perfect C-terp companion for your C compiler

C-terp/Microsoft
C-terp/Lattice
C-terp/Mark Williams
C-terp/XENIX
C-terp/Aztec
C-terp/C86

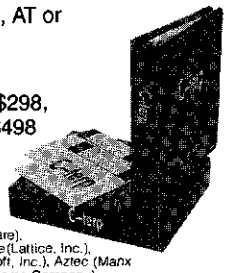
Link in all your compiler's functions, your own functions, add-on libraries, assembly routines, and data objects. Get instant access to everything in the C-terp interactive environment.

Only C-terp offers all this and more

- **Full K&R with common ANSI enhancements**
- **Source level interactive debugging**
- **Software paging for your big jobs**
- **Complete multi-module support**
- **Run-time pointer checking**
- **Unsurpassed reconfigurable screen editor**
- **Dual display and full graphics support**
- **Large model** ■ **Call-in**

ORDER C-terp TODAY (specify compiler)

C-terp runs on IBM PC, AT or compatibles.



Price:
MS-DOS 2.x and up - $298,
Xenix System V 286 - $498
MC, VISA, COD
30-day money-back
GUARANTEE

Trademarks: C-terp (Gimpel Software), C86 (Computer Innovations), Lattice(Lattice, Inc.), Xenix, Microsoft, MS-DOS (Microsoft, Inc.), Aztec (Manx Software), Mark Williams (Mark Williams Company), IBM (International Business Machines, Inc.)

GIMPEL SOFTWARE
3207 Hogarth Lane, Collegeville, PA 19426
(215) 584-4261

Number 11 on the Reader Service Card

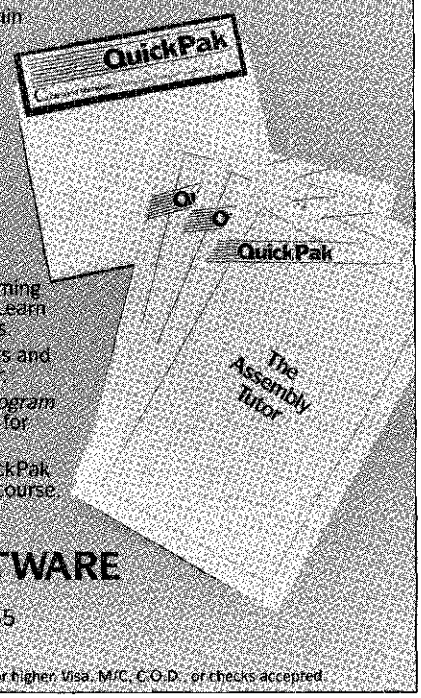
QuickBASIC just got quicker with QuickPak

QuickPak is a superb collection of enhancements, subroutines, and instructional material designed to help you get the most out of programming in BASIC.

- Powerful assembly language routines to give your programs more speed, more power, and full access to DOS and BIOS services. SORT all or part of a string array with one command! FIND any string or sub-string within an entire array *regardless of capitalization* — accepts wildcards. READ directories into your programs from any drive or path. READ/WRITE disk sectors — create your own DOS utilities! CLEAR or SCROLL any part of the screen. MANY, many more programs included.
- Professionally written QuickBASIC routines and functions. Powerful input routines for text, dates, and numbers. Menus, scroll bars, date/time functions, and much more.
- The Assembly Tutor — a complete guide to learning assembly language *from a BASIC perspective*. Learn how to create your own routines and extensions.
- Tips and Tricks book — packed with clever ideas and techniques to help *you* be a better programmer.

You get *all* this, all of the source code for *every program included*, and a thirty-day money back guarantee for only $69.00.

No royalties are required for using any of the QuickPak routines in your programs. Not copy protected, of course.



by
CRESCENT SOFTWARE
64 Fort Point Street
East Norwalk, CT 06855
(203) 846-2500

QuickPak requires Microsoft QuickBASIC or BASCOM, DOS 2.0 or higher. Visa, M/C, C.O.D. or checks accepted.

Number 83 on the Reader Service Card

PRODUCED 2023 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

★ NOW AVAILABLE FOR C ★ QUICKBASIC ★
★ IBM BASIC/BASICA ★ TURBO PASCAL ★

The Aspen Systems Subroutine Editor (**ASE**) you can call from your programs. Design your own screen layouts - - update in several windows simultaneously. **ASE** is:

- ★ **TRANSPORTABLE** - can be configured for any keyboard and almost any computer or language under MSDOS.
- ★ **VERSATILE** - customize input for any application.
- ★ **COMPLETE** - with full screen edit capabilities and a wide variety of automatic conversions.
- ★ **EASY TO USE** - data and screen layouts described in a single map.
- ★ **FULLY DOCUMENTED** - including examples in BASIC, PASCAL, FORTRAN, C, and even COBOL.
- ★ **AFFORDABLE-**

ASE is still only..... **$99**
DEMO (See what ASE can do for you!)..... **$ 3**

The Aspen Systems Subroutine Package (**ASP**) - the companion package for **ASE** - with many functions unavailable or difficult to perform in high level languages. Like **ASE, ASP** is:

TRANSPORTABLE ★ FULLY DOCUMENTED
EFFICIENT ★ AFFORDABLE

ASP includes 150+ subroutines: conversions; sorts; string, bit, date/time functions; decimal arithmetic, and **MORE**.

ASP is still only..... **$130**

ASE and **ASP** include support and demo programs.

Prices are PPD (continental USA).
Colorado Residents add 3%.

P. O. Box 1163
Grand Junction, CO 81502
(303) 245-3262
VISA/MasterCard accepted

ASPEN SYSTEMS

CP/M and MS-DOS are trademarks of Digital Research and Microsoft, respectively.

Number **86** on the Reader Service Card

80386
SOFTWARE DEVELOPMENT TOOLS

The Phar Lap 80386 Software Development Series:

386|ASM/LINK by Phar Lap (MS-DOS®) $495
Full-featured macro assembler and linker. Includes a debugger and 32-bit protected mode runtime environment.

80386 High-C™ by MetaWare (MS-DOS®) $895

80386 Professional Pascal™ by MetaWare (MS-DOS®) $895

UNIX™ and VAX/VMS® cross tools (call)

The wait for professional software development tools for the 80386 is over! Whether you are upgrading an existing IBM PC application to the '386, moving a mainframe application down to a PC, or writing the next PC best-seller, the Phar Lap 80386 Software Development Series is for you. It is an integrated line of products which provides everything you'll need to create and *run* 80386 protected mode applications under MS-DOS.

(617) 661-1510

Phar Lap Software, Inc. "The 80386 Software Experts"

60 Aberdeen Ave. Cambridge, MA 02138

Number **96** on the Reader Service Card

Software Tools
For Programmers & Non-Programmers

Get 'State of the Art' performance and save valuable time with these high quality utilities!

Opt-Tech Sort™

Opt-Tech Sort is a high performance Sort/Merge/Select utility. It can read, sort and write a file faster than most programs can even read the data. Example: 1,000 records of 80 bytes can be read, sorted and a new file written in less than 10 seconds (IBM XT). Opt-Tech Sort can be used as a stand-alone program or called as a subroutine to over 25 different programming languages.

All the sorting, record selection and reformatting facilities you need are included. A partial list of features includes: The ability to process files of any size. Numerous *filetypes are supported including Sequential, Random,* Delimited, Btrieve, dBASE II & III and many others. Up to 10 key fields can be specified (ascending or descending order). Over 16 different types of data supported. Powerful record selection capability allows you to specify which records are to be included on your output. Record reformatting allows you to change the structure of your output record and to output special fields such as record numbers for use as indexes.

MS-DOS $149. ★ NEW ★ Xenix $249.

★ NEW VERSION ★

On-Line Help™

★ NEW VERSION ★

On-Line Help allows you to easily add "Help Windows" to all your programs. On-Line Help is actually two help packages in one. You get BOTH Resident (pop-up) and Callable Help Systems.

The resident version allows you to add help to any system. Your Help System is activated when the "Hot Keys" that you specify are pressed. You can then chain between help windows in any manner you desire.

The callable version allows you to easily display help windows from your programs. A simple call to the help system makes the window appear. The original screen is automatically restored when the help window is cleared. On-Line Help is callable from over 20 different languages.

You have full control over the help window content, size, color and location.

MS-DOS $149. **Demo $10.** (apply toward purchase).

Scroll & Recall™

Scroll & Recall is a resident screen and keyboard enhancement. It allows you to conveniently scroll back through data that has gone off the top of your display screen. Up to 27 screens of data can be recalled or written to a disk file (great for documenting systems operations). Also allows you to easily recall and edit your previously entered DOS commands without retyping.

Scroll & Recall is very easy to use. It's a resident utility that's always there when you need it. MS-DOS $69.

Visa, M/C, AMEX, Check, Money Order, COD or Purchase Orders accepted.

To order or to receive additional information just call and receive immediate highly qualified attention!

Opt-Tech Data Processing
P.O. Box 678 — Zephyr Cove, NV 89448
(702) 588-3737

Number **16** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

EGA Data Control

by
Michael Abrash

This month I'm going to cover a lot of ground. First, I'll discuss the monitors the EGA can support and look into issues of EGA compatibility with other adapters. Next, I'll continue our detailed exploration of the data management capabilities of the GDC chip. Finally, I'll look at three EEGAs.

Color, Monochrome, and Compatibility

I've become aware that few people understand exactly what displays EGAs can drive and what adapters EGAs are compatible with. I'm going to do my best to remedy that here.

A standard EGA can drive a monochrome display, a color display, or an enhanced color display. Monochrome operation is simpler, so I'll start with that.

When driving a monochrome display, the EGA supports a text mode identical in appearance to that of the Monochrome Display Adapter (MDA), with characters in 9 x 14 boxes. The bit map is at B000:0000, just as with the MDA, and attributes can be (and are set up by the BIOS to be) interpreted in the same way as with the MDA. The EGA is BIOS-compatible with the MDA, but the registers of the EGA are only partially compatible. Most importantly, the cursor location register is MDA-compatible. The cursor start and stop scan line registers are not fully compatible, but they can be set reliably through the BIOS, which is compatible across the MDA, CGA, and EGA. Given the way in which the MDA is normally programmed, the EGA can be considered to be highly MDA-compatible. In particular, if registers other than the cursor location aren't set directly, the EGA is a fine MDA replacement. Since the MDA supports only one mode, mode 7, there is little reason to directly program MDA registers other than the cursor location, so the EGA can generally be substituted for the MDA.

When configured for monochrome operation, the EGA provides a graphics mode with a resolution of 640 x 350, with attributes of black, white, high-intensity, and blink. This mode is similar to the 640 x 350 color graphics mode except that two planes, rather than four, are used. Unfortunately, the EGA's monochrome graphics mode is not even remotely compatible with the graphics mode of the Hercules Graphics Card (HGC). It is possible to program an EGA to support the same bit map as the HGC does (someday I'll show how in this space). Since, however, the HGC has no BIOS support for mode switches, all software that supports the HGC programs the hardware directly; because the EGA is not fully register-compatible with the HGC, this will not work with the EGA. So, practically speaking, the standard EGA is compatible with the MDA but not the HGC.

Color EGA operation is more complex. The EGA can be configured to support only 200-scan-line display modes, or it can be configured to support both 200- and 350-scan-line modes. (In the configuration switch table for the EGA there appear to be more choices, but the only important choice is between 200 and 350 scan lines. There is no reason to power up an EGA in 40-column mode since it has no composite or RF output and the high-resolution emulation mode is identical to the 200-scan-line 80-column setting.) In 200-scan-line operation, the EGA has two new graphics modes (unsupported by the CGA), 320 x 200 in 16 colors and 640 x 200 in 16 colors. In 350-scan-line operation, the EGA has a 640 x 350 16-color graphics mode; additionally, in 350-scan-line operation text in 80 x 25 text mode is in an 8 x 14 box, rather than in the coarse 8 x 8 box of the CGA. High-resolution text is far superior to CGA text, but there is one important limitation—border colors other than black aren't supported in 350-scan-line modes (due to tight retrace timings). With that background out of the way, let's look at EGA emulation of the CGA.

In general, the EGA is only a fair Color/Graphics Adapter (CGA) replacement. The EGA supports all the display modes of the CGA, as well as all CGA bit maps at B800:0000. The EGA is also BIOS-compatible with the CGA, but, unlike the MDA's, the CGA's hardware is often programmed directly, and the EGA is not fully register-compatible. Most notably, the Color Select register, which is used to select among the CGA's palettes and background colors, is not present in the EGA. A BIOS function is provided to set these registers, but it supports only one of the two palette intensities provided by the CGA. Also, border colors in high-resolution modes aren't supported. In order to write a program that would run on both the CGA and EGA, you must avoid all accesses to registers other than the cursor location and start address registers, again including the cursor start and stop registers. This is possible, but the differences in color capabilities and color setting techniques between the two adapters mean that a program that would run on both adapters would be limited to two three-color palettes in graphics modes, with a black border only. In short, the EGA can be used to run BIOS-compatible CGA software, but the two boards are really quite different.

A summary of the standard EGA's compatibility with existing display adapters: CGA and MDA programs that do not access registers other than the cursor location and start address registers directly should run on the EGA. Bit-map compatibility is maintained by the EGA. HGC programs will not run on a standard EGA although it is possible to program an EGA to support an HGC-format bit map.

Many EGAs have extended CGA and/or HGC compatibility built in. These boards can create headaches for developers, for many of them are not really hardware-compatible with the

PRODUCED 2024 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

CGA or HGC. A common method of implementing compatibility is to generate an NMI (non-maskable interrupt) on attempted access to the CGA's (or HGC's) registers, with the NMI-handling routine forcing the CRT registers to settings that match the setting of the CGA mode register. This approach means that software that reprograms the 6845 CRT of the CGA (to get extra rows and columns, for example, or to smooth scroll) will not work on EGAs with NMI-based compatibility.

There are, incidentally, other problems with boards that provide downward compatibility—in particular that it can be difficult to determine what type of adapter a program should be trying to drive; for example, a program that looks for the string "IBM" in the EGA BIOS to see whether an EGA is installed (a common, although crude, method) will attempt to drive an EGA even if emulation is active and the program should be trying to drive an emulated CGA. One approach to this problem is letting the user select the display type at installation time.

As a result, the recommendation that developers leave registers other than the cursor location and start address registers alone and go through the BIOS holds even on CGA- and HGC-compatible EGAs. Bit maps are the same, so it's all right to go directly to display memory (fortunately, since the BIOS write and read dot functions are so slow as to be virtually useless). It's too bad that the BIOS doesn't provide full support for CGA palettes, though, and that border colors aren't supported in 350-scan-line modes.

Of course, if you're not interested in having your applications run on anything but an EGA (or anything but a CGA), or if you're willing to write your programs to specifically support a number of display adapters, you can go right to the hardware. I don't mean to say that you shouldn't use fast, nonstandard techniques when you have to—just be aware that as the world of display adapters expands, with CGAs, MDAs, HGCs, EGAs, EEGAs, and new display technologies that provide downward compatibility to varying degrees, it's going to be harder to support them all directly. So. . .where you can afford to, go through the BIOS; go directly to the hardware when you can't live without it.

There's one other compatibility issue concerning the EGA, and that's what displays the EGA can coexist with. The answer is that an EGA configured for color can be in the same system with an MDA (or HGC with up to 32K available), and an EGA configured for monochrome can be in the same system with a CGA. This is possible because a color EGA uses registers in the range 3C0–3DC and a monochrome EGA uses registers in the range 3B0–3CF, complementing the address ranges of the MDA and CGA, respectively. Two EGAs, one color and one monochrome, cannot coexist because both would use registers in the 3CX range. It is possible to jumper an EGA to set its address range to 2XX rather than 3XX, allowing two EGAs to be installed together without conflict; however, no current BIOS supports this alternate address range, so a good deal of work would be required in order to use two EGAs together in this fashion.

A Closer Look at the GDC

Last time, we looked at a simplified model of data flow within the GDC chips of the EGA, featuring the latches and ALUs. This month I'll expand that model to include the barrel shifter and bit mask, leaving only the set/reset capabilities to be explored.

EGA Data Rotation

Figure 1 shows the expanded model of GDC data flow. First, I'd like to look at the barrel shifter. (A barrel shifter is circuitry capable of shifting—or in the EGA's case, rotating—data an arbitrary number of bits in a single cycle.) The barrel shifter can rotate incoming CPU data up to seven bits to the right (toward the least significant bit), with bit 0 wrapping back to bit 7. Thanks to the nature of barrel shifters, this rotation requires no extra processing time over normal, unrotated EGA operation. The number of bits by which CPU data is shifted is controlled by bits 2–0 of GDC register 3, the Data Rotate register, which also contains the ALU function select bits.

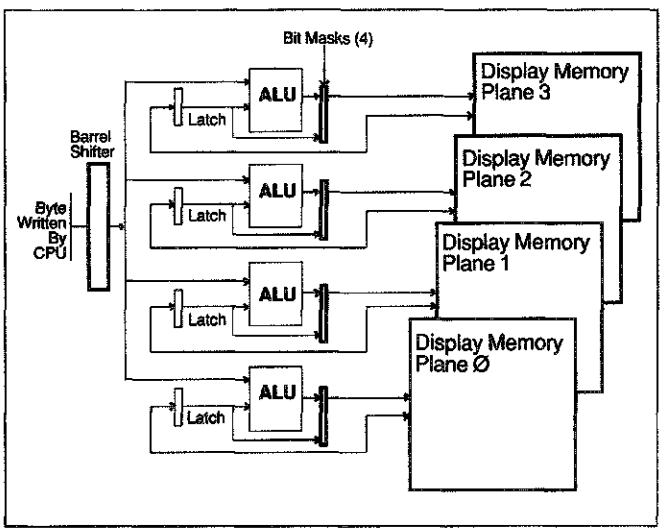


Figure 1: Data flow through the GDC (Set/Reset circuitry not shown)

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The barrel shifter is powerful, but it sounds more useful than it really is. This is because the GDCs can rotate only CPU data, a task which the CPU itself is perfectly capable of performing. Two OUTs (one to set the GDC index register and one to set the Data Rotate register) are needed to select a given rotation, and often it's easier and/or faster to simply have the CPU rotate the data of interest CL times than it is to set the Data Rotate register. If only the EGA could rotate latched data, there would be all sorts of useful applications for rotation, but, sadly, only CPU data can be rotated.

The drawing of bit-mapped text is one good use of the barrel shifter, and I'll demonstrate that application below. In general, though, don't knock yourself out trying to figure out how to work data rotation into your programs—it just isn't all that useful.

The Bit Mask

The EGA has a bit mask for each of the four memory planes. Figure 2 illustrates the operation of one bit of a bit mask. This circuitry occurs eight times in the bit mask for a given plane, once for each bit of the byte written to display memory. Briefly, the bit mask determines on a bit-by-bit basis whether the source for each byte written to display memory is the ALU for that plane or the latch for that plane. The bit mask is controlled by GDC register 8, the Bit Mask register. If a given bit of the Bit Mask register is 1, the corresponding bit of data from the ALUs is written to display memory, while if that bit is 0, the corresponding bit of data from the latches is written to display memory unchanged.

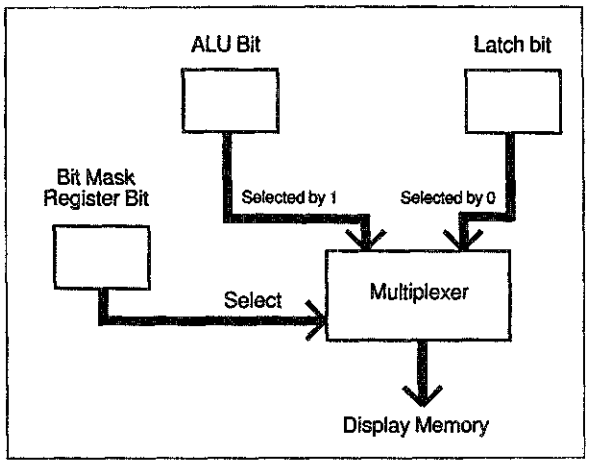


Figure 2: Operation of one bit of the Bit Mask register

The most common use of the bit mask is to allow updating of selected bits within a display memory byte. This works as follows: The byte of interest is latched; the bit mask is set to preserve all but the bit or bits to be changed; the CPU writes to display memory with the bit mask preserving the indicated latch bits and allowing ALU data through to change the other bits. Remember, though, that it is not possible to alter selected bits in a display memory byte directly; the byte must first be latched by a CPU read, and then the bit mask can keep selected bits of the latched byte unchanged.

Listing 1 shows a program that uses the bit mask and data rotation capabilities of the GDC to draw bit-mapped text at any screen location. (The BIOS draws only characters on character boundaries; in 640 x 350 graphics mode the default font is drawn on byte boundaries horizontally and every 14 scan lines vertically.) Note that the 8 x 8 BIOS font is used; a pointer to this font is obtained through function 11h of INT 10h.

The bit mask can be used for much more than bit-aligned fonts. For example, the bit mask is useful for fast pixel drawing, such as that performed when drawing lines. At some point in the future, I'll demonstrate such applications. The example in Listing 1 is designed to illustrate the use of the Data Rotate and Bit Mask registers and is not as fast or as complete as it might be. The case where text is byte-aligned could be detected and performed much faster, without the use of the Bit Mask or Data Rotate registers and with only one display memory access per font byte, rather than four. Also, it would be faster to use a table lookup to calculate the bit masks for the two halves of each character than the shifts used in the example.

For another (and more complex) example of drawing bit-mapped text on the EGA, see John Cockerham's article, "Pixel Alignment of EGA Fonts," *PC Tech Journal*, January 1987. Parenthetically, I'd like to pass along John's comment about the EGA: "When programming the EGA, everything is complex." He's got a point.

Of Interest

I've gotten the opportunity to try out several EEGAs, with impressive results. If you've got a monitor capable of going up to 640 x 480 or higher, you should consider an EEGA, and if you're developing an application that needs higher resolution (what application doesn't?), you should certainly take a long look at supporting one or more EEGAs. As we'll see, all EEGAs are not alike; each of the three I'll look at has unique features.

Before I begin, I'd like to make it clear that the following is not a review in any conventional sense. I'm not trying to cover the boards in every detail because I'm not trying to help you decide which to buy. By the same token, I'm not trying to blanket the field—you won't find comparison charts of 32 EGAs here. What I am trying to do is help you decide whether you should support the special features of these boards, or whether one of them might help you solve special hardware needs. You might think of topics discussed under the "Of Interest" heading as overviews of advanced display technology that I think developers should know about.

The VEGA Deluxe from Video-7 (550 Sycamore Drive, Milpitas, CA 94035, (408) 943-0101) is the follow-up to the extremely popular VEGA, one of the boards that fueled the first great boom in the EGA market. The VEGA Deluxe is built around the Chips & Technology EGA chip set, which has proved to be highly IBM EGA-compatible. The VEGA Deluxe supports resolutions of 640 x 480 and 752 x 410 although the only means of accessing these resolutions in the base package is via Windows drivers. (Late note: Video-7 indicates that other drivers are on the way and that developer's information is available.) The VEGA Deluxe also features full CGA and HGC compatibility. Given Video-7's stature in the EGA market, it would make sense for any developer who is going to support EEGAs to include the VEGA Deluxe.

The EGA 2001/PLUS, from Ahead Systems (1977 O'Toole Ave., Suite B105, San Jose, CA 94131, (408) 435-0707) is also built around the C & T chip set. The 2001/PLUS offers full CGA and HGC compatibility and adds a nice touch—HGC compatibility on color monitors. A 640 x 480 mode is supported (I propose this as a minimum requirement for a board to be considered an EEGA), as well as 720 x 396, which provides IBM S3G graphics compatibility. 80 x 66/45/43/34/25 and 132 x 44/28/25 text modes are provided; these modes provide IBM 3278/79 MOD 2–5 compatibility, as well as compatibility with many popular PC extended spreadsheet and text editing modes. The board is a compact, clean half-card. In short, the

PRODUCED 2007 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

2001/PLUS is a well-conceived and executed board that covers a lot of ground, and it's worth looking at if your requirements exceed the standard EGA.

The EVA 480, from Tseng Laboratories (205 Pheasant Run, Newtown, PA 18940, (215) 968-0502) sports a number of extras. Like the other EEGAs, the EVA 480 offers 640 x 480 graphics and complete downward compatibility with the CGA and HGC, as well as 132 x 44/28/25 and 80 x 60/43/25 text and Windows, Autocad, and Halo drivers. Because it is built around Tseng Labs' proprietary ET2000-series chip set (which, incidentally, I participated in the design of), the EVA 480 sports two unique extras: a zoomable hardware window and faster memory access.

The EVA 480 features a built-in hardware window (in addition to the standard EGA split screen), which can start and stop on any scan line vertically and on any byte boundary horizontally. This window can display the contents of any area of EGA memory, allowing instant popups, for example. The window area can be zoomed by pixel replication in both directions in graphics modes and horizontally in text mode, by a factor of up to 8 times horizontally and 16 times vertically. This could be very handy for a developer who needs to blow up part or all of the screen quickly.

The EVA 480 also allows the CPU much greater access to display memory than does the standard EGA. The standard EGA allows one CPU access to display memory every four character clocks while the EVA 480 allows one access every character clock. During highly display-memory-intensive operations, the EVA 480 is noticeably quicker than other EGAs. For example, I ran ATPERF, *PC Tech Journal*'s 286 Performance and Compatibility Suite ("Out from the Shadow of IBM . . ." by Steven Armbrust, Ted Forgeron, and Paul Pierce, *PC Tech Journal*, August 1986), on a 10-MHz AT clone in order to determine how many wait states were being inserted by each of the three EEGAs I've discussed. Both the VEGA Deluxe and the EGA 2001/PLUS performed at the same speed, inserting 18 wait states on average—not surprising given that they're built around the same chip set. The EVA 480 inserted only 9 wait states, the minimum number of wait states an 8-bit device can cause during a 16-bit access in an AT. For programs that access display memory less frequently, the gain will be less, but the EVA 480 is clearly the faster EGA.

Tseng Labs is the first but not the only company to make a higher performance EGA chip set. Chips and Technologies is coming out with a chip set with all sorts of extensions to the EGA, including more frequent CPU access to display memory. I'll discuss this chip set in more detail next time.

Don't expect the relatively sluggish performance of the EGA to improve much beyond the EVA 480, though, no matter what chip sets appear. Improving CPU memory access is about all that can be done without losing EGA compatibility, and right now compatibility is essential. Moreover, the higher performance from improved CPU memory access barely balances the greater size of the bit maps in the EEGA's highest-resolution modes.

On Performance

I'm running short on space, so all I'm going to say about performance is to suggest that you read "High-Performance Software Analysis on the IBM PC" by Byron Sheppard (*BYTE*, January 1987). It looks to me like Sheppard's routines provide very good software-based performance analysis. To the extent that I've tried them out, they've worked well and given pretty much the results I'd expect (although I will say that figur-

Introducing

Quaid Analyzer

the tool that created CopyWrite

Now you can debug your own programs with a professional quality debugger - the one that unraveled every form of copy-protection used on the PC.

With the Quaid Analyzer, you can:

- ☐ See occurences of any interupt, with its meaning shown on the screen.
- ☐ View memory as text or instructions, scrolling as easily as you do with an editor.
- ☐ Run until a memory location or I/O port is changed.
- ☐ Protect your hard disk from accidental destruction.
- ☐ Analyze software without the source, even when it uses countermeasures to thwart tracing.
- ☐ See all stages of the boot load.

We kept the Quaid Analyzer off the market to avoid helping publishers with copy-protection. Now that copy-protection is gone, we can sell it to you.

The Quaid Analyzer is a software tool occupying 100K bytes. It runs on any IBM PC and most MS-DOS systems without hardware modification.

To order Call **(416) 961-8243**

Quaid Analyzer $99 U.S.

All orders shipped at our expense within a day. All major credit cards accepted.

or return coupon to:
45 Charles St. East
Third Floor, Dept. 603
Toronto, Ontario. M4Y 1S2

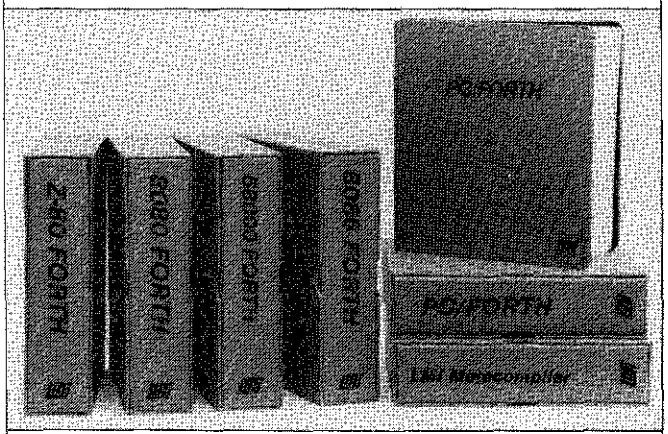
Payment method MC-Visa-Amex-Diners-Check
Card No. _____
Expiry Date _____
Name _____
Address _____
City/State _____
Phone No. _____
Signature _____

Quaid Software Limited

Ask about Disk Explorer the program that takes over where Quaid Analyzer leaves off.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

TOTAL CONTROL
with *LMI FORTH*™



For Programming Professionals:
an expanding family of compatible, high-performance, Forth-83 Standard compilers for microcomputers

For Development: Interactive Forth-83 Interpreter/Compilers
- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 400 page manual written in plain English
- Options include software floating point, arithmetic coprocessor support, symbolic debugger, native code compilers, and graphics support

For Applications: Forth-83 Metacompiler
- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, and 6303
- No license fee or royalty for compiled applications

For Speed: CForth Application Compiler
- Translates "high-level" Forth into in-line, optimized machine code
- Can generate ROMable code

Support Services for registered users:
- Technical Assistance Hotline
- Periodic newsletters and low-cost updates
- Bulletin Board System

Call or write for detailed product information and prices. Consulting and Educational Services available by special arrangement.

Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to: ***(213) 306-7412***

Overseas Distributors.
Germany: Forth-Systeme Angelika Flesch, Titisee-Neustadt, 7651-1665
UK: System Science Ltd., London, 01-248 0962
France: Micro-Sigma S.A.R.L., Paris, (1) 42.65.95.16
Japan: Southern Pacific Ltd., Yokohama, 045-314-9514
Australia: Wave-onic Associates, Wilson, W.A., (09) 451-2946

Number 63 on the Reader Service Card

ing actual jump times on the 286 appears at first cut to be black magic). Thanks to changes induced in the prefetch queue by the timing routines, results aren't necessarily accurate to a cycle, but then there's no way around that with a software solution. On quick examination, it looks to me as if Sheppard's done an excellent job.

Reader Feedback

Tom Heavey provided a solution to my problem of how to generate code that assembles at a specific address (for ROM, in my case). Tom's solution bypasses the Macro Assembler entirely; he suggests using a program called Link and Locate, from Systems and Software, Inc. ((714) 241-8650), in place of the standard MS DOS linker, with which it is compatible. According to Tom, Link and Locate adds a locate phase after the link phase, during which you can specify exactly where segments are to go.

Charles Quinton also provided a solution, but one that I can get to work only under certain circumstances. Charles points out that it's acceptable to have two relocatable values in an expression, so long as they combine to generate an absolute, so he suggests using

```
DB      2000h - ( $ - ProgramStart ) dup (0)
```

where ProgramStart is the offset of the first byte of the program. The assembler does indeed accept this syntax, and so long as ProgramStart is in the same module as the above DB statement, everything's fine. If, however, ProgramStart is in another module, too many bytes are inserted because "$" is relative to the start of the module, not to the start of the final linked program. Perhaps Charles meant this technique to be used only in single-module programs, or perhaps I'm missing something. Charles, can you write and let us know?

John Navas II provided a similar solution. Instead of the start of the program, however, he subtracts the segment name from the location counter. So far as I can tell, this operates exactly as Charles Quinton's approach does. John also sent in answers to two other MASM problems I've discussed here; they arrived too late to make it into this article, but I'll run them next time.

Thanks to all of you.

Next Time

We've almost completed our tour of GDC data flow—only the set/reset features remain. I'll cover these odd but useful features next time. If there's room, I'll begin to strike out into the more specialized read and write modes of the EGA.

See you then. □

Michael Abrash is a Senior Software Engineer for Orion Instruments of Redwood City, CA, a manufacturer of PC-based instrumentation and microprocessor development systems.

Code follows.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Code from Abrash, EGA Data Control

1

```
; Program to illustrate operation of data rotate and bit mask
;  features of Graphics Data Controller.  Draws 8x8 character at
;  specified location, using EGA's 8x8 ROM font.  Designed
;  for use with modes 0Dh, 0Eh, 0Fh, and 10h.
; Assembled with MASM 4.0, linked with LINK 3.51.
; By Michael Abrash, 2/8/87.
;
stack   segment para stack 'STACK'
        db      512 dup(?)
stack   ends
;
EGA_VIDEO_SEGMENT       equ     0a000h  ;EGA display memory segment
SCREEN_WIDTH_IN_BYTES   equ     044ah   ;offset of BIOS variable
FONT_CHARACTER_SIZE     equ     8       ;# bytes in each font char
;
; EGA register equates.
;
GDC_INDEX       equ     3ceh    ;GDC index register
GDC_ROTATE      equ     3       ;GDC data rotate/logical function
                                ; register index
GDC_BIT_MASK    equ     8       ;GDC bit mask register index
;
dseg    segment para common 'DATA'
TEST_TEXT_ROW   equ     69      ;row to display test text at
TEST_TEXT_COL   equ     17      ;column to display test text at
TEST_TEXT_WIDTH equ     8       ;width of a character in pixels
TestString      label   byte
        db      'Hello, world!',0       ;test string to print.
FontPointer     dd      ?               ;font offset
dseg    ends
;
; Macro to set indexed register INDEX of GDC chip to SETTING.
;
SETGDC  macro   INDEX, SETTING
        mov     dx,GDC_INDEX
        mov     ax,(SETTING SHL 8) OR INDEX
        out     dx,ax
        endm
;
cseg    segment para public 'CODE'
        assume  cs:cseg, ds:dseg
start   proc    near
        mov     ax,dseg
        mov     ds,ax
;
; Select 640x350 graphics mode.
;
        mov     ax,010h
        int     10h
;
; Set driver to use the 8x8 font.
;
        mov     ah,11h  ;EGA BIOS character generator function,
        mov     al,30h  ; return info subfunction
        mov     bh,3    ;get 8x8 font pointer
        int     10h
        call    SelectFont
;
; Print the test string.
;
```

2

```
        mov     si,offset TestString
        mov     bx,TEST_TEXT_ROW
        mov     cx,TEST_TEXT_COL
StringOutLoop:
        lodsb
        and     al,al
        jz      StringOutDone
        call    DrawChar
        add     cx,TEST_TEXT_WIDTH
        jmp     StringOutLoop
StringOutDone:
;
; Reset the data rotate and bit mask registers.
;
        SETGDC  GDC_ROTATE, 0
        SETGDC  GDC_BIT_MASK, 0ffh
;
; Exit to DOS.
;
        mov     ah,4ch
        int     21h
Start   endp
;
; Subroutine to draw a text character in a linear graphics mode
;  (0Dh, 0Eh, 0Fh, 010h).
; Font used should be pointed to by FontPointer.
;
; Input:
;  AL = character to draw
;  BX = row to draw text character at
;  CX = column to draw text character at
;
;  Forces ALU function to "move".
;
DrawChar        proc    near
        push    ax
        push    bx
        push    cx
        push    dx
        push    si
        push    di
        push    bp
        push    ds
;
; Set DS:SI to point to font and ES to point to display memory.
;
        lds     si,[FontPointer]        ;point to font
        mov     dx,EGA_VIDEO_SEGMENT
        mov     es,dx                   ;point to display memory
;
; Calculate screen address of byte character starts in.
;
        push    ds      ;point to BIOS data segment
        sub     dx,dx
        mov     ds,dx
        xchg    ax,bx
        mov     di,ds:[SCREEN_WIDTH_IN_BYTES]   ;retrieve BIOS
                                                ; screen width
        pop     ds
        mul     di      ;calculate offset of start of row
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

EGA Data Control, Continued

3

```
        push    di      ;set aside screen width
        mov     di,cx   ;set aside the column
        and     cl,0111b ;keep only the column in-byte address
        shr     di,1
        shr     di,1
        shr     di,1    ;divide column by 8 to make a byte address
        add     di,ax   ;and point to byte
;
; Calculate font address of character.
;
        sub     bh,bh
        shl     bx,1    ;assumes 8 bytes per character; use
        shl     bx,1    ; a multiply otherwise
        shl     bx,1    ;offset in font of character
        add     si,bx   ;offset in font segment of character
;
; Set up the GDC rotation.
;
        mov     dx,GDC_INDEX
        mov     al,GDC_ROTATE
        mov     ah,cl
        out     dx,ax
;
; Set up BH as bit mask for left half,
; BL as rotation for right half.
;
        mov     bx,0ffffh
        shr     bh,cl
        neg     cl
        add     cl,8
        shl     bl,cl
;
; Draw the character, left half first, then right half in the
; succeeding byte, using the data rotation to position the character
; across the byte boundary and then using the bit mask to get the
; proper portion of the character into each byte.
; Does not check for case where character is byte-aligned and
; no rotation and only one write is required.
;
        mov     bp,FONT_CHARACTER_SIZE
        mov     dx,GDC_INDEX
        pop     cx      ;get back screen width
        dec     cx
        dec     cx      ; -2 because do two bytes for each char
CharacterLoop:
;
; Set the bit mask for the left half of the character.
;
        mov     al,GDC_BIT_MASK
        mov     ah,bh
        out     dx,ax
;
; Get the next character byte & write it to display memory.
; (Left half of character.)
;
        mov     al,[si]         ;get character byte
        mov     ah,es:[di]      ;load latches
        stosb                   ;write character byte
;
; Set the bit mask for the right half of the character.
```

4

```
;
        mov     al,GDC_BIT_MASK
        mov     ah,bl
        out     dx,ax
;
; Get the character byte again & write it to display memory.
; (Right half of character.)
;
        lodsb                   ;get character byte
        mov     ah,es:[di]      ;load latches
        stosb                   ;write character byte
;
; Point to next line of character in display memory.
;
        add     di,cx
;
        dec     bp
        jnz     CharacterLoop
;
        pop     ds
        pop     bp
        pop     di
        pop     si
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        ret
DrawChar        endp
;
; Set the pointer to the font to draw from to ES:BP.
;
SelectFont      proc    near
        mov     word ptr [FontPointer],bp       ;save pointer
        mov     word ptr [FontPointer+2],es
        ret
SelectFont      endp
;
cseg    ends
        end     start
```


PRODUCED 2023 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

YOUR SYSTEM'S KEY COMPONENT

The Only Magazine By And For Advanced Micro Users.

At last there is a magazine that brings you the strictly technical but practical information you need to stay up-to-date with the ever changing microcomputer technology . . . *Micro/Systems Journal*. *Micro/Systems Journal* is written with the needs of the systems integrator in mind—the individual who's involved in putting together the hardware and software pieces of the microcomputer puzzle.

In each issue of *Micro/Systems Journal* you'll find such useful and progressive articles as:

- Interfacing to Microsoft Windows
- Unix on the PC
- 80386 Programming
- High Resolution PC Graphics
- Using 80286 Protected Mode
- Multiprocessing and Multitasking

You'll get the hands-on, nuts and bolts information, insight and techniques that *Micro/Systems Journal* is famous for . . . in-depth tutorials, reviews, hints . . . the latest information on computer integration, networks and multi-tasking, languages, and operating systems . . . hard-hitting reviews.

To start your subscription to *Micro/Systems Journal* write to *Micro/Systems Journal*, 501 Galveston Dr., Redwood City, CA 94063. You'll receive a full year (6 issues) of *Micro/Systems Journal* for just $20, and enjoy the convenience of having M/SJ delivered to your doorstep each month. Don't wait . . . subscribe today!

MICRO/SYSTEMS JOURNAL
501 Galveston Drive, Redwood City, CA 94063

YES! Please start my one-year subscription (6 information-packed issues) to Micro/Systems Journal at the low price of just $20.

☐ Bill Me ☐ Payment Enclosed ☐ Please Charge My:

☐ Visa ☐ M/C ☐ American Express

Card # _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

4002

Number **105** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Modula-2 for Pascal Lovers

by
Tom Swan

Much has been written about whether Modula-2 will replace Pascal—but not much has been said about why it hasn't. Despite predictions, the language remains a wallflower even though most experts agree that Modula-2 is far more than just Pascal in a pretty new dress. Why then is it taking so long for this newest of the structured debutantes to come out?

As one of many Pascal lovers who want to become better acquainted with Modula-2, I believe I know some of the reasons holding back this late bloomer. For one, Pascal is familiar. It's the kid next door—a steady date. Most of us have Pascal compilers and love the one we're with. Why risk playing the field?

But is the problem mere reluctance to change partners? Or, does Modula-2's unpopularity indicate deep flaws in its character? To answer these questions, I began about a year ago to collect Modula-2 information. Since then, I've read several tutorials, studied dozens of articles, and talked with many Pascal and Modula-2 programmers. I've run test programs through four compilers on as many different personal computers. The following, then, is not a tutorial, and it's not a review. Rather, it is a look at Modula-2 from a head-over-heels Pascal lover's eyes.

Pascal Criticisms

Pascal's faults are like painful sore thumbs that throb all day long. And the more hands-on experience you have, the more you bump into the old bruises, aggravating already inflamed sore spots. But even if few people agree on the cure, most recognize the following symptoms among Pascal's ills.

- No short-circuit Boolean expressions
- Fixed size arrays and strings
- Lack of constant expressions
- No ELSE in CASE statements
- No static variables
- No low-level machine access
- No separate compilation
- No multi-tasking ability

Some Pascal compilers fix a few of these faults with different degrees of success. Of them all, Turbo Pascal deals more successfully with these problems—undoubtedly a major reason behind Turbo's popularity. It does what most experts agree Pascal ought to do.

But even in Turbo Pascal, the lack of short-circuited Boolean expressions is particularly exasperating. The problem occurs because Pascal always evaluates all parts of an expression. For example, let's say you want to test whether an array index is in range while, at the same time, you test the value of an array element. You write

```
VAR a : ARRAY[1..10] OF INTEGER;
    i : INTEGER;

IF ( ( 1 <= i  )   AND
     ( i <= 10 ) ) AND
   ( a[i] < 100  )
   THEN doSomething;
```

and get into trouble. The first AND clause checks if i is in the range 1..10. The second AND clause checks if the value at a[i] is less than 100. But if the first clause is FALSE, the second could be ignored. In fact, unless i is in bounds, indexing a[i] can produce a fatal range error and, therefore, must not be allowed to occur.

In Modula-2, this same example is correct as written. If the index is not in the proper range, the second AND clause is not evaluated, avoiding the range error.

The complaint about fixed size arrays is well-taken but overblown. If you're willing to use pointers, you can easily simulate dynamic arrays and strings in Pascal. To its credit, though, Modula-2 provides a generic ARRAY data type for procedure parameters to which you pass arrays of unspecified size. Inside the procedure, HIGH(a) returns the highest array index. Although this is a needed improvement, Turbo Pascal has a similar ability with its untyped procedure parameters.

The lack of constant expressions is more serious. In Pascal, you might write something like this:

```
CONST
     Max = 10;
     MaxLess1 = 9;
```

These two constants are loosely related, a bad situation that can lead to bugs. If you later change Max to 20 but forget to change MaxLess1 to 19, the program fails. Modula-2's constant expressions avoid this problem. By using expressions, you tie the constants together so that if you change the first, the other automatically adjusts:

```
CONST
     Max = 10;
     MaxLess1 = Max-1;
```


PRODUCED 2023 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The criticisms about no ELSE in CASE statements, no static variables, and no low-level machine access probably are exaggerated. Most modern Pascal compilers have these features. Certainly, Turbo Pascal does.

The lack of separate compilation and multi-tasking are major problems with Pascal. Although Turbo has overlays, some say they are no substitute for separate compilation. ProPascal, an ISO standard compiler, does generate separate object modules, which easily link to modules created by other means. UCSD Pascal meets these complaints halfway with its separately compiled UNITS and segment procedures. It also has limited multi-tasking abilities. Of course Modula-2 supports separate compilation by definition—it wouldn't be Modula-2 if it didn't generate separate program modules.

While few Pascal compilers offer multi-tasking, it's a standard Modula-2 feature and one of its major stengths. You can write true co-routines, which execute independently of each other. But judging from a three-inch stack of articles and listings I've collected from various magazines, multi-tasking is one of Modula-2's least-used features. Undoubtedly, this will change as programmers move more powerful systems onto their desks.

Major Modula-2 Criticisms

The major Modula-2 barbs are that it's a wordy language, its I/O is unnecessarily difficult to use, and the compiler runs more slowly than Pascal. And many people complain that its improvements to Pascal seem more arbitrary than helpful.

It is wordy. Because you have to declare all imported procedures and data types from other modules, you can expect to do a lot of typing. In text form, Modula-2 programs tend to be somewhat longer than Pascal equivalents. This is not a language for hunters and peckers although a good keyboard macro program will help.

I/O is difficult and unfriendly. Each data type requires its own procedure for both input and output. To write a character, a string, and a number and start a new line requires these four statements:

```
WriteChar( ch );
WriteString( s );
WriteCardinal( n );
WriteLn;
```

The argument goes that, even though you have to write so many separate statements, a Pascal compiler generates identical code for this:

```
WriteLn( ch, s, n );
```

But, as a friend put it, "Why should I have to do what Pascal does for me? Isn't that one of the reasons you use a compiler in the first place?"

That's a good point and one of the principal reasons behind Modula-2's struggles—the perception that you have to work harder to write programs in the language. At the same time, I understand the logic behind separate I/O calls for different data types. Because modules handle all I/O, you can redesign even low-level statements such as WriteChar and WriteLn. As a contrast, there's no way to replace Turbo Pascal's WriteLn statement—except, of course, to design a new procedure with a different name. And that means you would have to revise existing programs to take advantage of your new programming, which—at least in theory—you would not have to do if you decide to replace Modula-2's standard I/O modules.

Another valid criticism concerns speed. The compiler's prime feature—modularity—requires it to read dozens of separate files for each typical compilation. This can slow things down. Modula-2 proponents argue that with separate compilation, total compile times go down even though individual compile times increase. But the truth is that no Modula-2 compiler can hope to match Turbo Pascal's compilation speed. And that's another major reason Modula-2 is fighting its battle uphill.

What Have They Done to My Compiler, Ma?

People seem unable to decide whether Modula-2 is a new language or a better Pascal. This indicates that Wirth made a mistake by renaming it while insisting at the same time that this isn't a new language but, rather, an improved successor. Founding *BYTE* editor Carl Helmers agrees: "Maybe if Wirth had called it Modular-Pascal, we'd all be using it today."

If you think that a specious argument, consider what happened to Personal Software. They changed their name to VisiCorp and went invisibly out of business. And, as another example, do you think that Ford labels one of its models "Mustang" because the automobile's design really is related to that lovely '66 mare of the same name? No. The name suggests continuity. Maybe programmers would feel more comfortable with Modula-2 if they perceived it to be the close relative to Pascal it claims to be.

Despite its new name, however, Modula-2's parentage is no mystery. It shines over Pascal in modularity, multi-tasking, and low-level access. It repairs most of the Pascal criticisms I mentioned earlier, adding short-circuited Boolean expressions, dynamic arrays, constant expressions, and static variables. Un-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

fortunately, the changes don't stop there and some of its minor improvements seem purely syntactic, even arbitrary. For example, all structured statements must now end with END. You can no longer write

```
WHILE n > 10 DO
      n := n - 1;
```

Instead you have to write

```
WHILE n > 10 DO
      n := n - 1
END;
```

And compound statements in Pascal such as

```
WHILE n > 10 DO
   BEGIN
      WriteLn( n );
      n := n - 1
END;
```

now look like this:

```
WHILE n > 10 DO
   WriteLn( n );
   n := n - 1
END;
```

Such changes disappoint Pascal lovers. What is wrong with BEGIN? I like the way I can sit back and see the structure of a Pascal program. The BEGINs and ENDs jut out at me like boulders along a ridge line. I don't get that same sense of order from Modula-2.

The addition of a new keyword, ELSEIF, also seems unnecessary. To do the same in Pascal requires a space as in ELSE IF. So what's the advantage? I thought one of Pascal's strong points was its English key words. There's no ELSEIF in English.

And why oh why is this language case sensitive? WriteLn, writeLn, Writeln, and writeln are four different identifiers. This is clearly a concession to the compiler writers and nothing but an annoying hindrance to programmers. I can believe no argument that claims case sensitivity helps me to write better computer programs.

Another wide-spread criticism of Modula-2 is for its elimination of braces as comment markers. In Pascal, there are two comment symbol pairs, { } and (* *). You can nest either inside the other, which lets you comment out large sections of code without worrying about whether that code contains other comments.

But Modula-2 requires (* and *) for all comments. Braces now surround sets as they do in traditional math—the probable reason that these characters no longer surround comments. But how can requiring me to type four characters instead of two be an improvement? The argument in favor of this change points out that you can now nest (* and *) to any level. Great. Now I can nest comments seven levels deep. Just what I always wanted.

Are you Ready for Modula-2?

If you know Pascal, you'll have little trouble learning Modula-2. But if you're new to Pascal, you might find Modula-2 confusing. In fact, another main criticism of the language comes from educators who continue to prefer Pascal for programming courses. Modula-2, they say, is inappropriate for teaching and more difficult for beginners to learn than Pascal. Perhaps this factor is at the top of the list of reasons for Modula-2's slow rise.

Be warned that subtle differences between Modula-2 and Pascal make switching back and forth from one to the other as difficult as hopping mossy stones in a mountain stream. This is a new trouble for me. I frequently shift among 8086, 6502, and Z80 processors—and from one computer to another. But when I go from Modula-2 back to Pascal, I often forget whether to use BEGIN or to add a procedure name after END or to use braces or brackets for sets. When you use both languages, their similarities get in the way and their minor differences are more annoying than helpful.

But to decide if you agree with my opinions, you need to at least learn the fundamentals of Modula-2. For those who don't know where to begin, several suggestions for a quick self-study course follow.

How to Learn Modula-2

You can certainly learn Modula-2 from books, and one book outshines all for Pascal lovers: Richard Gleaves's *Modula-2 for Pascal Programmers*, Springer-Verlag. You can read this 145-page book in one or two evenings, and it may be the only tutorial you'll need. It's a fine little book.

Modula-2 Programming by Ian Kaplan and Mike Miller, published by Hayden (now Howard W. Sams), is a good introduction to Modula-2 for those who don't have a lot of Pascal experience. The treatment is simple but complete.

Niklaus Wirth's two books *Programming in Modula-2* (Springer-Verlag) and *Algorithms and Data Structures* (Prentice-Hall) are disappointments. The first in its third corrected version is a must for serious programmers—it's the reference bible to the language. But the second you can do without. A remake of the popular *Algorithms + Data Structures = Programs*, this Modula-2 version lacks the shine, humor, and good writing of its earlier Pascal edition. It's badly in need of editing and redesign, its language is stilted, and its logic is hard to follow. You'd be better off taking the Pascal edition and converting the programs to Modula-2 yourself.

Modula-2/86

Of course, you can't learn a language by reading only books. You'll need a compiler, too. Logitech's Modula-2/86 is the leading compiler and probably the best choice. You can purchase a stripped version for as little as $89 or a complete package with debugger and utilities for $199. Library sources are another $99. Other extras include a window package ($49), a make utility ($29), and a Turbo Pascal to Modula-2 translator ($49).

Altogether, if you order all the fixin's, you'll receive 13 diskettes and need from 1.5 to 3.0 megabytes of disk space to hold the compiler, linker, run-time support files, and utilities. Except for learning the language, don't even think about using Modula-2 on a floppy disk system. You need a hard disk for serious work. And a large RAM-disk won't hurt.

The Editor

Modula-2/86 comes with a multi-window editor, MOD. There's no limit to the number of files you can edit at once, but Logitech suggests a practical maximum of six.

MOD understands Modula-2. You can perform quick syntax checks without calling the compiler—a real help especially while learning the language. You can also run compiled programs without exiting to DOS. You can run the compiler and linker from the editor or execute them from the DOS command line. This lets you use your favorite text editor to edit program

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

sources—a thoughtful feature.

On the negative side, MOD is not WordStar-compatible. The command sequences and function key arrangements are poorly arranged and difficult to memorize. Also, it does not save a backup copy of your work file—a serious omission.

The Compiler and Linker

To compile a Modula-2 program, you can run the compiler from the editor or type M2 COMP on the DOS command line. You can also execute the fully linked compiler M2C—an optional purchase—if you have at least 512K memory.

As with the compiler, you can run the linker from the editor or type M2 LINK from DOS. Or, you can execute the optional fully-linked linker, M2L. However you run it, the linker generates load files, ending in .LOD. To run a load file, you type M2 <filename>, which loads the run-time support package M2, which in turn loads and runs your program. After compiling and linking, you can convert .LOD files to stand-alone .EXE format with a supplied utility.

The Manual

Logitech's manual is well written and organized. There's a short tutorial that introduces Modula-2, but to learn the language, you'll still need the other references I mentioned.

Installation is quick and painless, and Logitech provides several batch files to automate practically everything. Step-by-step instructions get you going quickly. I had little trouble running test programs.

Modula-2's Future

The future looks good for Modula-2. But don't expect Pascal to die out anytime soon. Compilers such as Turbo Pascal already fix many of the complaints that inspired Niklaus Wirth to improve the language. And rumors are that Turbo Pascal 4.0—possibly due out this year—will offer separate compilation and will come with a multi-window editor.

In the confusion over whether to switch to Modula-2, maybe we Pascal lovers could use some advice from a different kind of expert. In an old movie James Cagney, telling a co-worker why he was leaving work for a date with his new love, said, "If you want a flower to blossom, you've got to cultivate it." Something like that.

By that reasoning, perhaps Modula-2 isn't popular because programmers haven't spent enough time cultivating the language. Even so, maybe someday it will meet expectations and win our affection. But before that happens, programmers will have to accept Modula-2 as a help-mate and not as the antagonist they perceive it to be now. □

Tom Swan is the author of several books including Pascal Programs for Data Base Management *and* Mastering Turbo Pascal *published by Howard W. Sams, Inc.*

MYSTIC PASCAL

Fastest Compiler on Earth!

- integrated programming environment with editor, incremental compiler, direct mode
- true multi-tasking for up to 100 program sections
- powerful color graphics with pan and zoom on 20 screens
- 10 virtual consoles
- 8087 and non-8087 floating point math
- up to 640K of code, data or stack
- run programs in memory or create OBJ modules on disk
- 30 day money back guarantee
- requires IBM PC or compatible with 320K

Add $2 for shipping in US or Canada. Overseas add $10 for disk & manual or $5 for disk-only. Payment must be in US funds on a US bank. Check/Visa/MC accepted. COD is $3 extra.

505/757-6344
Mystic Canyon Software
P.O. Box 1010
Pecos, New Mexico 87552

$40 Disk & Complete Printed Manual

$24 Disk-Only Version with abbreviated manual on disk

Number 95 on the Reader Service Card

TURBO PASCAL GENERATOR

THE GTP PROFESSIONAL MODEL

Generate error-free Turbo Pascal source code for:

DATA BASES MENUS
SCREENS REPORTS

Build complete, working programs in minutes!

FEATURES
- Indexed Data Bases
- Multiple Screens
- Automatic Updating
- Built-in Edits
- Retrieval Facility
- Automatic File Build
- Quick Screen Handling
- Speedy DB Access
- Context Sensitive HELP
- Full Keyboard Support

EASY
1. Paint-the-screen
2. Define fields & calculations
3. Generate, compile & RUN

FLEXIBLE
- 100% Turbo Pascal
- Modify in Pascal under Turbo's Editor
- Extensions are easy with 130 page Programmer Reference manual
- 100 documented source code routines included

NOT Copy Protected NO Royalty Fees NO Run-time Library Required

Requires: IBM PC (100% compatible) 256K RAM 2 Disk Drives
PC-DOS 2.0+ Turbo Pascal 3.0 from Borland International

PRICE $200.00
CALL FOR QUANTITY DISCOUNTS
VISA/MC — Check — Money Order — No COD or Purchase Orders
Texas Residents: add $12.25 Sales Tax
Outside US & Canada: add $25.00 Air Postage and make payment by credit card or money order in US Funds.

A E F
ALLEN, EMERSON & FRANKLIN
P.O. Box 928
Katy, TX 77492
(713) 391-8570

Number 53 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C Programmer's Toolbox

Productivity Enhancement Tools

CFlow Generate a map of your program's subroutine hierarchy. Supports an unlimited number of program source listings.

CPrint Beautify your program listings, making them consistant and easy to read.

PMon Monitor your program execution and system calls, determining where the most time sensitive/critical sections are. Does not require any modification to an existing program.

***** And more.... All from one source....

$79.95 30 day money back guarantee
CA residents add applicabile sales tax

MMC AD Systems™
Box 360845
Milpitas, California 95035
(408)-263-0781

Number 94 on the Reader Service Card

NOW!

386 C and Pascal for MS-DOS

MetaWare Incorporated announces the *first* available C and Pascal compilers that generate

protected-mode 80386 code

for running on any 80386 machine that runs MS-DOS (e.g., the Compaq Deskpro 386 or the IBM Personal System/2 Model 80). The compilers are functionally identical to our well-respected 8086/286 MS-DOS **High C**™ and **Professional Pascal**™ compilers, but now you can get them generating 80386 code.

There's no reason to wait! Industry leaders such as ANSA and Fox Software are already converting their 16-bit database products to 32-bit protected mode, getting increases in speed and functionality. Don't wait years for Microsoft's 386DOS — your competition will have a big jump on you!

Expand your application to the large 32-bit address space and the full 32-bit registers of the 80386. Contact MetaWare for your 80386 software solution today!

(408) 429-6382, telex 493-0879.

MetaWare™
INCORPORATED
903 Pacific Avenue, Suite 201 • Santa Cruz, CA 95060-4429

Number 45 on the Reader Service Card

dBASE Programmers

are you ready?

dB→c

"C" IS READY FOR YOU!

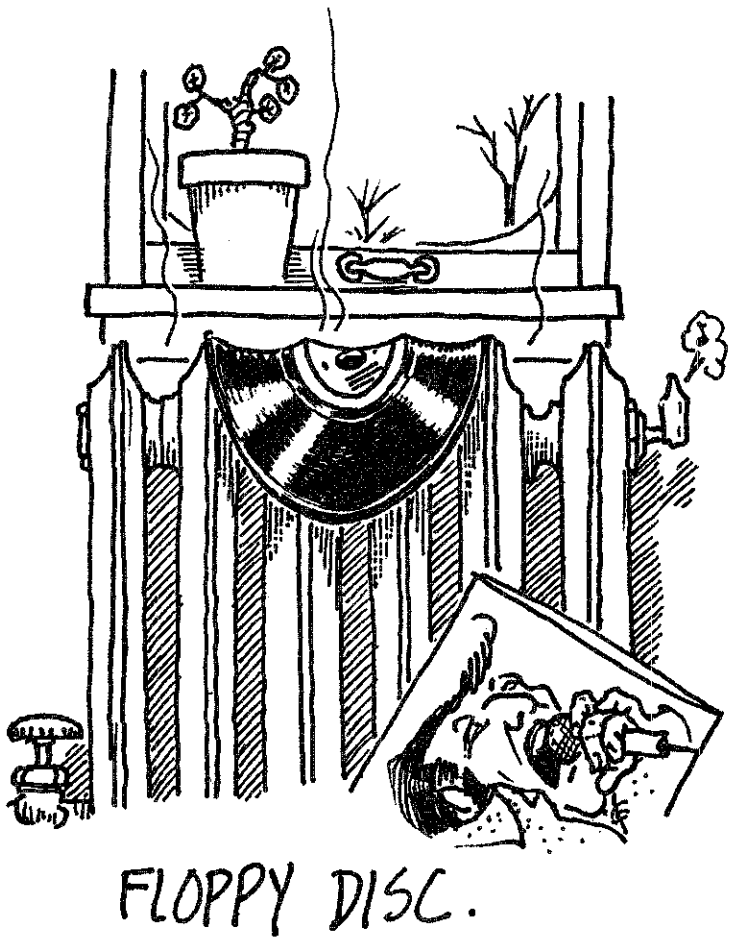
dB2c Offers:
- Complete implementation of dBASE III+ functions and commands in C source code.
- Interfaces for Lattice's dBC and Faircom's c-tree.
- Complete source code
- Support for the following C Compilers: Microsoft, Lattice, Aztec, DeSmet, and Instant-C.
- The freedom and peace of mind that only source code can offer.
- Works on color, monochrome, and EGA monitors.

dB2c Toolkit $299.

DAVID J. MARSH

Call or Write:
SOFTWARE CONNECTION, INC.
POB 712, Ely, MN 55731
(218) 365-5097

Number 93 on the Reader Service Card



PRODUCED 2007 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Dr. Dobb's C Tools

Dr. Dobb's Toolbook of C

This authoritative reference contains over 700 pages, including the best C articles from Dr. Dobb's Journal along with new material. You'll find hundreds of pages of valuable C source code, including a complete compiler, an assembler, and text processing programs.

Toolbook of C Item #005 $29.95

Small-C Handbook and Small-C Compiler

This compiler and handbook provide every you need for learning how compilers are constructed, and for learning C at its most fundamental level. You'll find a discussion of assembly language concepts and program translation tools, and of how to generate a new version of the compiler. Full source code is included. Please specify MS/PC-DOS or CP/M: Kaypro, Osborne, Apple, Zenith Z-100 DS/DD, 8" SS/SD.

CP/M Compiler & Handbook Item #006B $37.90
MS/PC-DOS Compiler & Handbook Item #006C $42.90

Small-Tools: Programs for Text Processing

This package of Small-C programs performs specific, modular operations on text files. It is supplied as source code with full documentation. With the Small-C Compiler, you can select and adapt these tools to meet your needs. Please specify MS/PC-DOS or CP/M: Kaypro, Osborne, Apple, Zenith Z-100 DS/DD, 8" SS/SD.

Small-Tools Item #010A $29.95

Small-Mac: An Assembler for Small-C

This package includes: a simplified macro facility, C language expression operators, object file visibility, descriptive error messages and an externally defined instruction table. Source code and documentation is included. CP/M only. Please specify: Kaypro, Osborne, Apple, Zenith Z-100 DS/DD, 8" SS/SD.

Small-Mac Item #012A $29.95

Small Windows: A Windowing Library for Small C

Small-Windows is a complete windowing library for Small-C. The package contains: 18 video functions written in assembly language; 7 menu functions that support both static and pop-up menus; and 41 window functions, including functions to clean, frame, move, hide, show, scroll, push, and pop windows. A file directory facility illustrates the use of the window menu functions and provides file selection, renaming and deletion capability. Two test programs are also included. For PC/MS-DOS systems only. Documentation and full source code is included.

Small-Windows Item #109 $29.95

Special Packages

CP/M **C** Package
Save Over $27!

Receive: Dr. Dobb's Toolbook of C, The Small-C Handbook and Small-C Compiler, Small-Mac Assembler, and Small-Tools Text Processing Programs. Only $99.95!

CP/M Package Item #005A $99.95

MS/PC-DOS **C** Package
Save $22!

Receive: Dr. Dobb's Toolbook of C, The Small-C Handbook and MS/PC-DOS Addendum, Small-C Compiler, Small-Tools Text Processing Programs and Small Windows. Only $109.95!

MS/PC-DOS Package Item #005W $109.95

To Order:

Return this order form with your payment to M & T Books, 501 Galveston Dr., Redwood City, CA 94063.

ORDER FORM

Name _____

Address _____

City _____ State _____ Zip _____

Item #	Description	Price

Please indicate MS/PC-DOS or CP/M. For CP/M specify: Apple, Osborne, Kaypro, Zenith Z-100 DS/DD, 8" SS/SD.

Subtotal _____

CA residents add sales tax _____ % _____

Add $2.25 per item for shipping _____

TOTAL _____

☐ Check Enclosed. **Make Payable to M & T Publishing Inc.**

Charge my ☐ VISA ☐ M/C ☐ Amer. Exp.

Card # _____ Exp.Date _____

Signature _____

4022

call TOLL-FREE 800-533-4372 (In CA call 800-356-2002)

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Optimizing Turbo Pascal

Borland's Compuserve SIG recently hosted a month-long seminar on optimizing Turbo Pascal. Here are some of the proceedings.

Initializing a Data Structure

From Joe Schrader (Sysop) 76703,4161
To All Seminarees

Neil Rubenking has developed this technique for initializing a complex data structure without using a configuration file or run-time disk I/O:

```
type
   FileHeaderType = record
                      { complex record structure }
                    end;
var
   FileHeader : FileHeaderType;
 { or alternatively }
   P : ^FileHeader;


procedure DefaultHeader; EXTERNAL 'default.hdr';
{ This procedure is never called, but Turbo is
tricked into reading in the data at compile time.
}

procedure Initialize;
{ At runtime no disk I/O (which is usually
the most time consuming operation)
is required to init FileHeader. }
begin
...
  Move(Mem[CSEG:Ofs(DefaultHeader)],
  FileHeader, SizeOf(FileHeader));
 { or alternatively }
 P := Ptr(CSEG,Ofs(DefaultHeader)); end;
 { Initialize }
...
```

Tighter Code

From Tom Neff 76556,2536
To Turbo Programmers

Here are a few ways to improve the tightness of Turbo's code.

1. Use integer instead of byte variables. Turbo manipulates integers almost all the time anyway, no matter what you code. The byte type just adds extra instructions onto each fetch and store operation. Change from byte to integer variables on a big program sometime and look at the code you save.

Don't forget enumerated types and subranges when you do this. That nice textbook example,

```
VAR today : (Monday, Tuesday, Wednesday, Thursday, Friday)
```

slaps you with a byte type for "today." Better to use

```
CONST Monday=0; Tuesday=1; Wednesday=2 etc...
VAR today : Integer;
```

and skip the enumerated range checking in favor of tighter code. The same thing goes for

```
VAR line : 1..25;
```

This gives you a byte, too. Use an integer and do your own checking.

2. Never use constant SET values; they are built dynamically at runtime every time. Declare a structured SET constant (pre-initialized CSeg variable) if you must.

3. The old warhorse: Don't use Length( ); use Ord(s[0]) whenever possible. It saves some code.

A Compiler Patch

From Bela Lubkin 76703,3015
To All

Try this patch for the TURBO.COM 3.01A PC DOS release (first instruction = JMP 2D7C):

```
debug turbo.com
nturbox.com
a12D
 or ax,ax
 jz 137
 es:mov [bx],ax
 jmp 8BAE
 sub cx,3
 sub word ptr [5A0],3
 sub word ptr [5A6],3
 jmp 134

a8BAB
 jmp 12D

w
q
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The patch hooks into the code generator at the point that generates JMP instructions. If the JMP being generated has no effect, i.e., is a JMP <next instruction>, it prevents it from being generated. This eliminates unneeded JMPs at the beginning and end of procedures.

It will not cause problems with things like Inline($E9/$00/$00); or Const X: Array [1..3] Of Byte=($E9,0,0); because it hooks into the JMP generator, not the generic bytestream generator.

I've tried to throw the book at it, and I haven't been able to get it to break. Normal code, overlays, externals, inlines, chain files, Extender. . . they all work.

Reply
From Clark Walker 76010,346 To Bela Lubkin

Bela: I tried it in a program with quite a few FORWARDs and didn't run into any problems. □

FORTRAN PROGRAMMERS

Looking for the right PC FORTRAN language system? If you're serious about your FORTRAN programming then you should be using F77L - LAHEY FORTRAN.

"Lahey's F77L FORTRAN is the compiler of choice. It's definitely a 'Programmers FORTRAN,' with features to aid both the casual and the professional programmer. . . F77L compiled the five files in a total of 12 minutes, which was 4 times as fast as MS FORTRAN and an astounding 6 times as fast as Pro FORTRAN." - PC Magazine

Compare the features and performance of other PC FORTRANs with F77L and you will find that F77L is clearly the superior product.

- Full Fortran 77 Standard (F77L is not a subset)
- Popular Extensions for easy porting of mini and mainframe applications
- COMPLEX*16, LOGICAL*1 and INTEGER*2
- Recursion - allocates local variables on the stack
- IEEE - Standard Floating Point
- Long variable names - 31 characters
- IMPLICIT NONE
- Fast Compile - Increases productivity
- Source On Line Debugger (Advanced features without recompiling)
- Arrays and Commons greater than 64K
- Clear and Precise English Diagnostics
- Compatibility with Popular 3rd Party Software (i.e. Lattice C)
- Easy to use manual
- Technical Support from LCS

- *NEW FEATURE - NAMELIST*

F77L - THE PROGRAMMER'S FORTRAN

$477.00 U.S.

System Requirements: MS-DOS or PC-DOS, 256K, math coprocessor (8087/80287)

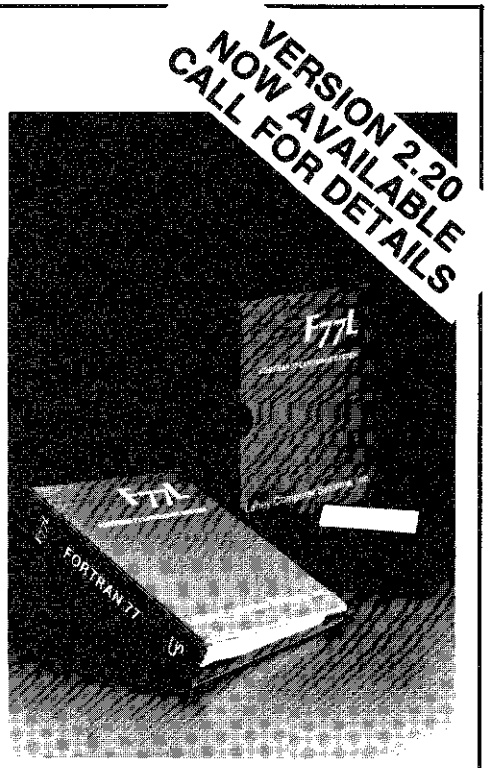
FOR MORE INFORMATION: (702) 831-2500

Lahey Computer Systems, Inc.
P.O.Box 6091
Incline Village, NV 89450
U.S.A.

International Dealers:
England: Grey Matter Ltd., Tel: (0364) 53499
Denmark: Ravenholm Computing, Tel: (02) 887249
Australia: Computer Transitions, Tel: (03) 537-2786
Japan: Microsoftware, Inc., Tel: (03) 813-8222

SERVING THE FORTRAN COMMUNITY SINCE 1967

MS-DOS & MS FORTRAN are trademarks of Microsoft Corporation Pro FORTRAN refers to Professional FORTRAN a trademark of International Business Machines.



Editor's Choice
- PC Magazine

Number **44** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Proposed ANSI C Language Standard, Part 7

by
Rex Jaeschke

©*Rex Jaeschke, 1987. All rights reserved.*

In writing this and the column in the last issue of *Programmer's Journal* ("The ANSI C Language Standard, Part 6"), I have used Microsoft C V4, Lattice C V3.0, and Wizard C V3, each of which claims to be tracking the proposed ANSI Standard—in any case, they all implement some form of function prototypes, the subject of this column and the preceding. These compilers will be referred to as MSC, LATC and WIZC, respectively, throughout the article. I have also been working with C-terp 2.30b (Lattice V3 variant) and Instant-C V2.2 (December 1986). These will be referred to as CT and IC, respectively. Other DOS C compilers are known to support function prototypes. These include Whitesmiths V3 and High-C from MetaWare.

Functions with No Argument Lists

Because prototypes were invented by the ANSI Committee, most existing C code does not use them; therefore, new-style function declarations must be able to coexist with old-style declarations if existing code is not to be broken immediately. An old-style function declaration is considered to be a prototype without an argument list; for example,

```
int f();
```

declares that function **f** has return type **int** and that no information is known about the argument list, so the compiler can not perform checking. The same situation occurs when you call a function that has never been declared. In the absence of a declaration, a function is deemed to be of return type **int**, and the compiler constructs a prototype for it. This prototype contains no argument list and is identical to the one shown above.

To indicate that a function has no arguments and to allow the compiler to police this, the format

```
int f(void);
```

should be used instead. (The new keyword **void** has three possible uses—that is, it is overloaded—as a function type, as a pointer type, and in function prototypes as shown here.) All three compilers tested supported the use of **void** in prototypes as shown.

Given that the ellipsis notation disables argument list checking, you might think that the following declarations are identical—neither provides any argument information:

```
int f();
int f(...);
```

As it happens, the second declaration is illegal according to the Draft ANSI Standard. (The ellipsis can only follow a comma, and since there is no first argument, a comma would be illegal.) The MSC *C Language Reference Manual* (see page 66—the Note and the paragraph before it) seems to indicate that this is allowable, yet it is not. The discussion about the ellipsis on that page applies only to an ellipsis after a comma, *not* to an ellipsis on its own. Depending on how you read it, the note on page 66 is misleading.

Prototypes and Argument Widening

Another (potential) benefit of prototypes is that they may be used to bypass the default widening rules when dealing with function arguments. (Whether or not a prototype allows the default widening rules to be overridden is implementation-defined.) In the following example,

```
f()
{
        extern void g();
        char c;
        short s;
        float f;

        ...
        g(c, s, f);
}
```

the **char c** and **short int s** are both widened to **int**, and the **float f** is widened to **double** before being passed to the function **g**. In function **g**, the argument declarations could be

```
void g(c, s, f)
char c;
short s;
float f;
{
        ...
}
```

However, if the prototype for function **g** were

```
void g(char c, short s, float f);
```

and it were in scope for both the calls to **g** and the definition of **g**, the compiler could bypass the default argument widening rules allowing the conversion overhead to be omitted. This is of particular use to programmers who use **float** instead of **double** and on systems where a subroutine call is needed for argument widening and "unwidening." They no longer need

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

to pay the price of **float/double/float** conversions. (Note that the current **math.h** routines still require **double** arguments, though, since all existing code expects widening to occur during function calls. However, the set of math library function names with a suffix of f and l is reserved for future use by the Standard so that versions of these routines can be provided that take and return float and long double values, respectively.)

While it can be messy, it is possible to have some arguments in a list widened while others in that same list are not, by using the ellipsis specifier. The ellipsis notation causes argument type checking to cease such that any arguments that actually follow will go unchecked. These unchecked arguments are widened as if no prototype were currently in scope.

New-style Function Definitions

Given that the prototype for strcmp is

```
char *strcmp(const char *s1, const char *s2);
```

you might expect that the actual definition of **strcmp** would be

```
char *strcmp(s1, s2)
const char *s1;
const char *s2;
{
        ...
}
```

While this format would be correct, it is made obsolescent by the Standard. The Standard often regards as obsolescent those features that are currently common practice but for which an alternate (replacement) mechanism has been provided. Programmers who have a strong incentive to write conforming programs should replace function definitions such as this with

```
char *strcmp(const char *s1, const char *s2)
{
        ...
}
```

This new-style function definition format combines the argument list names and their declarations in a Pascal-like manner. The simple reason for this is uniformity—the definition now exactly matches that of the corresponding prototype.

Similarly, the use of function declarations without argument lists as for **f( )** above, is obsolescent in requiring that the special identifier void, or some other prototype format, be used to insure conformity with future versions of the Standard.

While WIZC supports the new-style function definitions, MSC, LATC, CT, and IC do not.

Function main

There can be no arguments in a prototype for function **main** because the Standard permits **main** to have either no arguments or two arguments. And given that it is unlikely for a user program to call **main** itself, this is not a problem, just an idiosyncrasy of the Standard. Of course, the meaning of and type of any arguments beyond the first two are implementation-defined. A common third argument is **envp**, the environment pointer. This is an array of pointers to **char**, where each pointer points to some environment string. Numerous DOS C compilers implement this capability although to be strictly conforming, you should use the library function **getenv** instead.

The ANSI Rationale

The following statements are taken from the Rationale Document (X3J11/86-152) that accompanies the ANSI C Standard. This document explains some of the rationale that went into the Standard-making process. Section 3.9.3 (Function declarators) states,

> The characterization of the use of the "old-style" function declarations and definitions—that is, the traditional style not using prototypes—signals the Committee's intent that the new prototype style should eventually replace the old style.
>
> The case for the prototype style. . . is that the new syntax addresses some of the most glaring weaknesses of the language. . . , that the new style is superior to the old style on every count.
>
> It was obviously out of the question to remove syntax used in the overwhelming majority of extant C code, so the Standard specifies two ways of writing function declarations and definitions. Characterizing the old style as obsolescent is meant to discourage its use and to serve as a strong endorsement by the Committee of the new style. It confidently expects that approval and adoption of the prototype style will make it feasible for some future C Standard to remove the old style syntax.

To Convert or Not to Convert?

If you have a significant investment in C code already, should you upgrade to a new (and ANSI-like) C compiler? The answer to that question can vary considerably. Because the correct use of prototypes can add a significant amount of comfort to your

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

compilations, getting that capability alone may justify the conversion. However, you should note that newer compilers have changed in (possibly) subtle ways, and once ANSI-conforming compilers become available, the conversion effort may be even greater. The good news is that most compiler vendors will continue to support many of their old habits in the interim. But yes, you will have to answer that question eventually.

If you do choose to add prototypes to existing code, you may want to check out Pre-C, a LINT-like tool from Wizard Systems. A command-line switch allows you to extract function prototypes from input source code files that use the old-style function definitions. You can then place these prototypes in a header called, say, **prototyp.h** and **#include** that in all the places where those functions are called and defined. The MSC compile-switch /Zg does the same thing.

Note that most compilers supporting prototypes for library routines have both the old-style and the new-style declarations in their headers. The new style declarations are included if a special macro name is defined—in MSC this is LINT_ARGS. Check your compiler manual and its headers for details.

Future Meetings

The schedule for future ANSI meetings is

September 14-18, Boston, Massachusetts
December 7-11, Phoenix, Arizona. □

Rex Jaeschke is an independent computer consultant, writer and lecturer. He is also a member of the X3J11 C Standard's Committee, Co-founder and Editor of The C Journal, *and the C language Editor of* The DEC Professional.

If you would like to comment on or question any aspect of the Proposed ANSI C Standard, write to

Rex Jaeschke
2051 Swans Neck Way
Reston, VA 22091

Rex will try to address your concerns in future columns as space permits.

FREE SOURCE CODE!

VITAMIN C
It's good for your system!

FREE SOURCE CODE!

Vitamin C Difference

With **Vitamin C,** your applications come alive with windows that explode into view! Data entry windows and menus become a snap. Vitamin C's **open ended design** is full of "hooks" so you can "plug in" special handlers to customize most routines. Of course, Vitamin C **includes all source code FREE,** with no hidden charges. *It always has.*

Windows

Create windows with one easy function. Vitamin C automatically takes care of complicated tasks like saving and restoring the area under a window.

Options include titles, borders, colors, pop-up, pull-down, zoom-in, scroll bars, sizes to 32k, and more. Unique built-in feature lets users move and resize windows at run-time!

Data Entry

Flexible dBase-like data entry and display routines feature protected, invisible, required, and scrolling fields, picture clause formatting, full color/attribute control, selection sets, single field and full screen input, and unlimited validation via standard and user definable routines.

High Level Functions

Standard help handler provides context sensitive pop-up help messages any time the program awaits key strokes. So easy to use that a single function initializes and services requests by opening a window, locating, formatting, displaying and paging through the message.

Multi-level MacIntosh & Lotus style menus make user interfaces and front ends a snap. Menus can call other menus, functions, even data entry screens quickly and easily.

Text editor windows can be opened for pop-up note pads and general purpose editing. Features include insert, delete, word wrap, justify, cut, paste, search, and more!

VCScreen

With VCScreen and Vitamin C working together, you'll reach a new level or productivity you can't reach with a function library alone!

VCScreen speeds development even more! The interactive screen editor actually lets you draw input, output and constant fields, headings, boxes, lines, even a window for your froms to run in.

VCScreen generates readable C source code ready to "plug in" to your application and link with Vitamin C.

Guarantee

Better than a brochure. More than a demo disk. If you're not satisfied, simply return the package within 30 days and receive a full refund of the purchase price.

Vitamin C $225.00
Includes ready to use libraries, tutorial, reference manual, demo, sample and example programs, and quick reference card; for IBM PC and compatibles. Specify compiler and version when ordering.

Vitamin C Source . . . . . FREE*
**Free with purchase of Vitamin C.*

VCScreen $99.95
Requires Vitamin C and IBM PC/XT/AT or true compatible.

Shipping $3 ground, $6 2-day air, $20 overnight, $30 overseas. Visa and Master Card accepted. All funds must be U.S. dollars drawn on U.S. bank. Texas residents add 7¼% sales tax.

(214) 245-6090

creative PROGRAMMING
Creative Programming Consultants, Inc.
Box 112097 Carrollton, Texas 75011

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Scientific Desk®

A Problem Solving Environment in Mathematics and Statistics

For programmers, hundreds of callable Fortran subroutines use the most advanced techniques; includes a graphics library; available on many computer types.

For immediate results, Problem Solvers designed by experts require no programming. Documents, tutorials and examples make work easy.

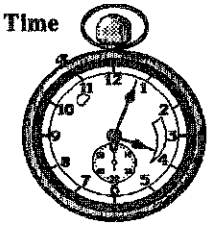
Includes

Arithmetic, Error Analysis	Special Functions
Linear Algebra (Eigenanalysis)	Interpolation
Solution of Non-linear Equations	Optimization
Differentiation and Integration	Differential Equations
Integral Transforms (ffts)	Data Fitting
Statistics	Data Handling

C. ABACI, 208 ST. MARY'S STREET, RALEIGH, NC 27605 (919) 832-4847
PRICE: $330 to $1200 (site license)

Number **107** on the Reader Service Card

Along With Your Computer, Your Time is the Most Important Thing You Own. . .So Why Waste It?



Quilt Programmer Productivity Tools will help you manage your software projects and get control of your time!

SRMS™
Software Revision Management System

- Full Featured Revision Control System
- All Versions stored in a Single ASCII File
- Support for Unlimited Libraries
- Support for all programming languages
- Allows you to use your current compilers and editors without conflict
- Windowing Shell interface to simplify use
- MERGE facility to consolidate different development paths easily, while pointing out conflicting areas
- Full audit trail tracking and reporting on all library components
- Handles big programming projects easily
- Full DOS Pathname and Environment Variable Support
- Requires DOS 2.1+, 224 K, F/H Disk

SRMS Version 3.0.....$185

QMAKE™
Intelligent Program Generation Utility

- Controls the rebuilding of even the most complex systems
- Relieves the developer of remembering which modules need to be rebuilt based on recent changes, how to rebuild them, and in what order to rebuild them
- Works with most compilers, assemblers, and linkers
- Supports full macro definitions, UNIX make-file compatability, recursive invocations, and command line parameters
- Interfaces completely with SRMS, providing you with a complete set of productivity tools to handle any size project
- Requires DOS 2.1+, 128K F/H Disk

QMAKE Version 1.2.....$99

SRMS + QMAKE $250

NEW ! TXT Tools NEW !

QSE - Quilt Text Stream Editor
QSRCH - Quilt File Search Utility (Like UNIX GREP)
QDIFF - Quilt Windowing File Difference Utility

TXTTOOLS Version 1.0..... $85

QUILT COMPUTING
7048 Stratford Road
Woodbury, MN 55125
(612) 739-4650

Volume Discounts and Dealer Inquiries Welcome

Number **46** on the Reader Service Card

LUCID, INC.

Learning UNIX/C In-Depth

"FOR THE CLARITY YOU EXPECT FROM TRAINING"

Ask about our HANDS-ON course for "C" Program Development Under MS/DOS"

For Free Brochure, Information or Registration Please Write or Call:
Alice Moss — Director

LUCID, INC.

260 Fifth Avenue
New York, N.Y. 10001
Suite 901

(212) 685-7106

Unix is a trademark of AT & T Bell Labs
MS-DOS is a trademark of Microsoft Corp.

Number **59** on the Reader Service Card

C CODE FOR THE PC

source code, of course

Panache C Program Generator	$150
QC88 C Compiler	$90
EMACS-like Editor	$75
TELE Kernel & Windows	$60
Make	$50
PC/MPX Multitasking Executive	$45
Coder's Prolog	$45
Biggerstaff's System Tools	$40
Translate Rules to C	$30
ICON String Processing Language	$25
Bison (YACC clone) & PREP	$25
LEX	$25
XLT Text Translator	$20

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409
(512) 258-0785

Free shipping on prepaid orders MasterCard/VISA

Number **30** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Coping with ANSI C and Forcing File Updates

by
William J. Hunt

Coping with ANSI C

Writing programs that compile correctly for several versions of Lattice and Microsoft compilers can be a problem. Many useful ANSI C features such as function prototypes, the void data type, and library functions are not supported by older versions of the compiler. My solution to this problem is to define a header file myansi.h that patches up differences between compiler versions. In the version below for Lattice 3.00 and Microsoft 3.00, defining the symbol FPRO specifies that function prototypes are supported.

```
/* myansi.h  - defines ANSI C features my compiler supports */
/* This version for Version 3.00 or later - Lattice or Microsoft */
#ifndef MYANSI                  /* once only test */
#define MYANSI 1

#define FPRO 1                  /* function prototypes */
/* void type supported */
/* memcpy in library - supply function prototype */
char *memcpy(char *,char *,unsigned) ;

#endif                          /* end of once only test */
```

Whenever we need to declare function prototypes or return values, we include "myansi.h" and test to see if the FPRO symbol is defined. For example, we would declare the malloc function with

```
#include "myansi.h"             /* defines FPRO */
#ifdef FPRO
char malloc(unsigned) ;         /* use a function prototype */
#else
char malloc() ;                 /* return value only */
#endif
```

For older versions of Lattice and Microsoft compilers, the myansi.h file below leaves FPRO undefined so that function prototypes are not used. It defines the void data type which is not recognized by these compilers and the memcpy function.

```
/* myansi.h  - defines ANSI C features my compiler supports */
/* Version for Lattice 2.15 and older and MSC 2.03 and older */
#ifndef MYANSI                  /* once only test */
#define MYANSI 1

/* leave FPRO undefined - no function prototypes */

typedef int void ;              /* void type not supported by compiler *
                                   /* memcpy not in library */
#define memcpy(to,from,n)   movmem(from,to,n)

#endif                          /* end of once only test */
```

Forcing File Updates

When you perform file updates in a C program, you often want those updates to be actually written to a file immediately. Then if a power failure or system crash occurs, the update will be present and the file intact when the system is restarted. Part of the problem, though, concerns MS DOS—you need to force DOS to flush data and the file control structures DOS uses (the directory and the File Allocation Table).

I suggested closing and re-opening the file to force DOS to flush data and control information. Another member of the LCUG, Michael Crandell, suggested a quicker method. The MS DOS function call DUP (0x45) returns a second file handle for an open file. Closing the file using the second file handle forces DOS to flush any updates, and the original file handle may be used in further reads and writes without re-opening the file.

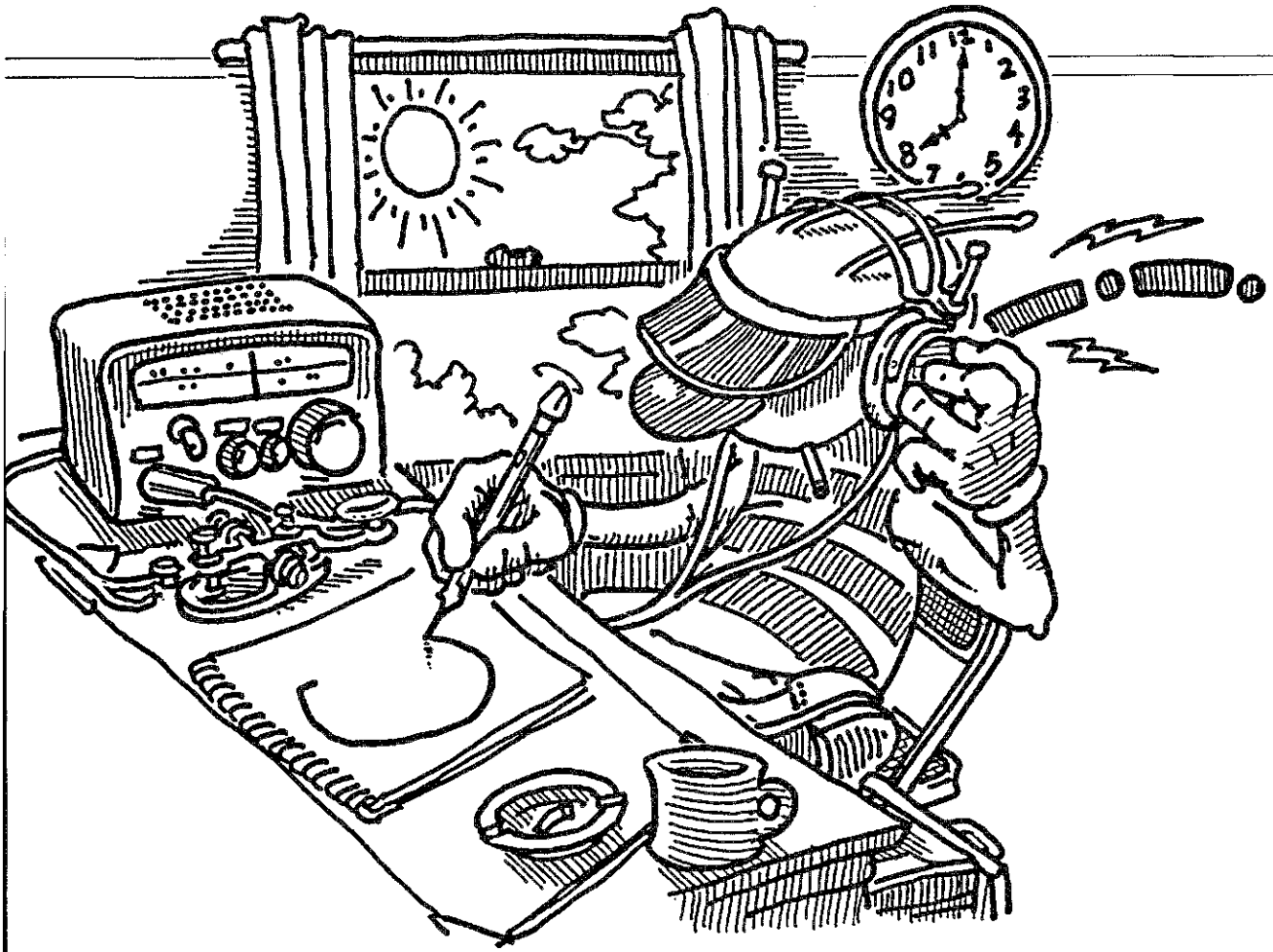
Lattice C Version 3.00, Microsoft C 3.00, and IBM C all provide the library functions for the DOS DUP call. The ioflush function below uses the dup function to get a second file handle and then closes the file using that handle. As the comment at the top of the source file suggests, the Microsoft/IBM versions require an additional header file.

```
/* ioflush.c - flush data, directory and FAT for a file */
/* version for Lattice Version 3 */
#include "fcntl.h"
/* For Microsoft version 3, add the following line */
/* #include "io.h" */

int ioflush(fh)                 /* force DOS to update file */
 int fh ;                       /* file handle returned by open/creat */
 {
   int new_fh ;                 /* file handle for duplicate */

   new_fh = dup(fh) ;           /* get a duplicate file handle */
   if( new_fh < 0 )             /* check for failure */
      return( -1 ) ;
   if( close(new_fh) < 0 )
      return( -1 ) ;
   return( 0 ) ;
 }
```

This ioflush function requires about 0.3 second on an XT hard disk. Closing and re-opening a file requires 0.1 to 0.3 second more. For files opened with an explicit path (c:\dir1\dir2\myfile) or files located in large directories, the difference may be even greater.

For Lattice C 2.15 and earlier versions, the ioflush2.c provides the same functions. The dos_fh function translates a library file number returned by open and creat into a DOS file handle using the undocumented data structure UFB and the

PRODUCED 2007 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



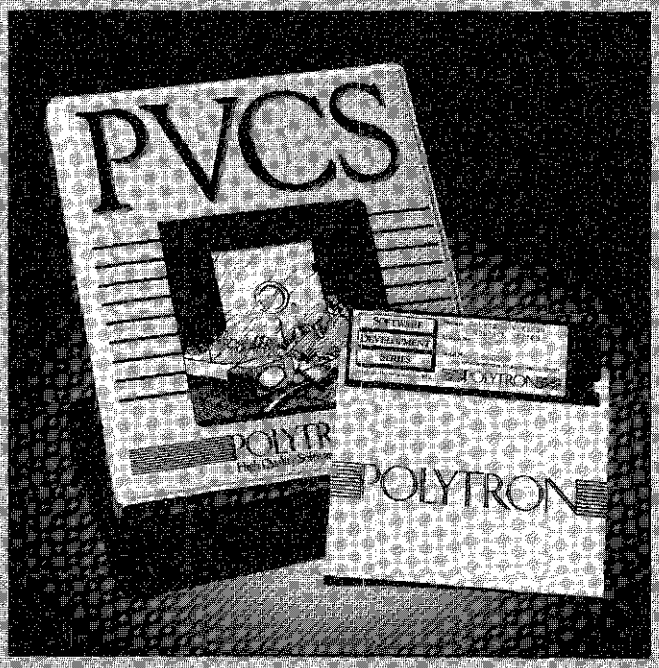
array of structures __ufbs. It also provides a dos__close function that uses a DOS file handle directly. Before version 3.00 the library did not include a dup function, so we provide that also. The source file below works with Lattice C Version 2.15—you will have to try it with earlier versions. □

William J. Hunt writes the Lattice C Users' Group *newsletter, where this material first appeared, and is also the author of* The C Toolbox, *published by Addison-Wesley. Readers may contact Bill at the following address:*

William J. Hunt
Lattice C Users' Group
P. O. Box 271965
Concord, CA 94527

```
/* ioflush2.c - Lattice 2.15 flush data function */
#include "stdio.h"
#include "fcntl.h"
#include "dos.h"
#include "ios1.h"
extern struct UFB _ufbs[]; /* UFB structures for each file */
#define DUP_CALL    0x45    /* DOS DUP call number */
#define CLOSE_CALL 0x3E     /* DOS CLOSE call number */
#define CARRY_FLAG   1      /* 8088 carry flag signals error return */

int ioflush(fh)             /* force DOS to update file */
 int fh ;                   /* file handle returned by open/creat */
 {
   int old_fh ;             /* DOS file handle for fh */
   int new_fh ;             /* DOS file handle for duplicate */

   old_fh = dos_fh(fh) ;    /* convert to DOS file handle */
   if( old_fh < 0 ) return( -1 ) ;
   new_fh = dos_dup(old_fh) ; /* get a duplicate file handle */
   if( new_fh < 0 ) return( -1 ) ;
   if( dos_close(new_fh) < 0 ) return( -1 ) ; /* close the file */
   else return( 0 ) ;
 }

int dos_fh(fh)              /* convert library file handle to DOS fh */
 int fh ;                   /* Lattice library file handle */
 {
   struct UFB *pufbs ;      /* points to library file structure */
   int dfh ;                /* DOS file handle */

   if( fh < 0 ) return( -1 ) ;
   pufbs = & _ufbs[fh] ;    /* get pointer to UFB structure */
   if( pufbs == NULL ) return( -1 ) ;
   dfh = pufbs->ufbfh ;     /* convert to DOS file handle */
   return( dfh ) ;
 }

int dos_dup(dos_fh)         /* duplicate a file handle */
 int dos_fh ;               /* an existing DOS file handle */
 {
   union REGS dos_regs ;

   dos_regs.h.ah = DUP_CALL ;  dos_regs.x.bx = dos_fh ;
   if( (intdos(& dos_regs,& dos_regs) & CARRY_FLAG) == 0 )
      return( dos_regs.x.ax ) ;
   else return( -1 ) ;
 }

int dos_close(dos_fh)       /* close a file handle */
 int dos_fh ;               /* a DOS file handle */
 {
   union REGS dos_regs ;

   dos_regs.h.ah = CLOSE_CALL ;  dos_regs.x.bx = dos_fh ;
   if( (intdos(& dos_regs,& dos_regs) & CARRY_FLAG) == 0 )
      return( 0 ) ;
   else return( -1 ) ;
 }
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



PVCS

The Most Powerful and Flexible Source Code Revision & Version Control System for PCs and Local Area Networks

Programmers, Programming Teams, Project Managers, Librarians and System Administrators Find PVCS Indispensable

When large and complex software programs and systems are being developed on personal computers, effective management of the revisions and versions becomes critical. The POLYTRON Version Control System (PVCS) simplifies this process and lets you effectively control the proliferation of code changes. We used UNIX SCCS and RCS as models. However, our own experience, and the input of hundreds of PC programmers has enabled us to significantly improve upon these models.

PVCS provides many powerful functions including:

* Storage & Retrieval of multiple revisions of text.
* Maintenance of a complete history of changes.
* Maintenance of separate lines of development using branching.
* Resolution of access conflicts.
* Merging of simultaneous changes.
* Modules can be retrieved by their own revision number, system version number, or specified date.
* Efficient disk storage. PVCS uses a very intelligent difference detection technique that minimizes the amount of disk space required to store a new version.

PVCS Maintains System Integrity

PVCS prevents system corruption that could ordinarily result from security breaks, user carelessness or malfunctions. The levels of security can be tailored to meet the needs of your project.

The Preferred Version Control System

PVCS is now being used in personal computer software development environments at hundreds of high-technology corporations including:

Hewlett-Packard	3 Com
GTE Data Services	IBM
Sperry	Texas Instruments
Westinghouse Electronics	ROLM Corp.
Nestar Systems	

Maintains Source Code Written In ANY Language.
Requires DOS 2.0 or higher. Compatible with the IBM PC, XT, AT and other MS-DOS PCs.

Only PVCS meets the needs of Independent Programmers and Corporations. Once you standardize on PVCS, the "Logfiles" used to track and monitor changes are interchangable between any PVCS product. You will receive full credit for your initial purchase if you upgrade to a higher-priced PVCS.

Personal PVCS — Offers most of the power and flexibility of the Corporate PVCS, but excludes the features necessary for multiple programmer projects. **$149**

Corporate PVCS — Offers additional features to maintain source code of very large and complex projects that may involve multiple programmers. Includes "Branching" to effectively maintain code when programs evolve on multiple paths (e.g., new versions for different systems, or a new program based on an existing program). Single user. **$395**

Network PVCS — Extends Corporate PVCS for use on Networks. File locking and security levels can be tailored for each project. 5-Station License $1,000. Call (503) 645-1150 for pricing on Licenses for more than 5 Stations. **$1000**

TO ORDER: VISA/MC 1-800-547-4000. Dept. No. 306. Oregon and outside US call (503) 684-3000. Send Checks, POs to: POLYTRON Corporation, 1815 NW 169th Place, Suite 2110, Dept. 306, Beaverton, OR 97006.

POLYTRON®

High Quality Software Since 1982

Number 26 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal For IBM PC Programmers
PROGRAMMER'S Journal

DIRECT MAIL ORDER

ORIGINAL DESIGN
T-SHIRTS

BORN TO PROGRAM PJT101

NOW!
Soft Wear from Programmer's Journal!
Only $9.95 each, Postage paid. Classic PJ cartoons & cover graphics. Reproduced as dynamic Multi-color imprints on high quality 50/50 cotton T-Shirts. In sizes S, M, L, XL. Born to program? Send for yours today!

RS-232 HEAD
PJT103

PARALLEL PORTS
PJT102

COMMON C BUG
PJT104

BACK ISSUES

•Vol. 1, No. 1: Norton batch files and Pascal I/O; Program style manual; COBOL on the IBM PC; Benchmarks; System analysis; BASIC modules; Rounding error.

•Vol. 1, No. 2: Defining character sets; C86 BIOS calls; Troubleshooting; Copy protection; Pascal code; Input routines; Writing an assembler.

•Vol. 1, No. 3: ASM dumb terminal; ASM tutorial routines; C source code; ASM graphics line drawing; Text editors reviewed; Utilities; Self-documenting programs; Text files to dBASE II; dBASE II PEEKS & POKES.

•Vol. 1, No. 4: Diskette directory via Pascal; MASM WHILEWEND; C/ASM interface; ASM stay resident routine; Structured programming; Software law; Pascal tutorial; Default values and variable names.

•Vol. 1, No. 5: Multi-tasking software review; Transporting C programs; Programmer's calculator; dBASE II bargraphs; Utilities; ROM BIOS equates; Pascal check program; Combining modules in Assembler; Structured programming in BASIC; Agreement outline.

•Vol. 1, No. 6: ASM printer configuration; DOS patch utility; Hi-speed graphics; Software ownership; Structured programming revisited; C and BASIC compared; Software protection; dWINDOW review.

•Vol. 2, No. 1: Freelance consulting; Sorting in C; Pascal text editor source code; CRT distress; Compiled BASIC; Assembly modules for C; Designing a relational database; Ultra Utilities; Punctuation and Style reviewed.

•Vol. 2, No. 2: Interview with Brian Kernighan; Understanding C through assembly language; Utility routines in C and assembly; Music in C; Automatic line number optimizing in BASIC; Assembly language tutorial; Full-screen input routines in BASIC; dBASE II printer control codes; Pascal tools reviewed; C programming support products.

•Vol. 2, No. 3: Software development environments; Creating a productive screen handling tool; Assembly tutorial; Generating graphics within windows; Compression and editing system for musical ideas; Hybridizing assembly and BASIC; Creating screens; Linking Assembler to Pascal and BASIC.

•Vol. 2, No. 4: Writing a custom ROM BIOS for PC DOS; RISC, Transputers, and mass storage; The future of disk protection; 8088 assembly macros; Random access files in BASIC; C utility routines you can buy; Using the PMATE macro language; Choosing a database manager; E/SP syntax-checking; Pascal editor reviewed.

•Vol. 2, No. 5: Developing the APX CORE Executive; From Assembler to C via a 211 byte .EXE maint(); Scroll control using BASIC and Assembler; Faster screen handling with BASIC and ASM; Random access files, Part 2; RTOOL, a PC-resident programmer's toolkit; dBkey reviewed; Pascal's pointer power.

•Vol. 2, No. 6: Periscope—building a hardware board for the IBM PC; Intel 80286 bugs by FNE; AT&T surprises all; Field Usage and pointer subtraction in C; Calendar functions for BASIC; Memory segmentation; Reduce your .EXE file sizes; Enhancing Debug; dBASE II to dBASE III? THINKTANK reviewed.

•Vol. 3, No. 1: November's Comdex; C—its uses and abuses; Using the C preprocessor; Memory segmentation examples; Index files, Part 1; Structured programming language; Periscope, a symbolic debugger.

•Vol. 3, No. 2: UNIX! (again); Artificial Intelligence; Doctor C's tips—pointers to functions; Calendar date routines in BASIC; Calculate record layouts, automatically!

•Vol. 3, No. 3: NAPLPS and the professional programmer; Instant-C; Printer control from a batch file; Repositioning the stars in Starfinder; Reducing EXE file size through data positioning; tools for better programming.

•Vol 3, No. 4: More UNIX stuff; C programming books, Part 1; C code excerpts, Part 1; How to cure the "dreaded locked keyboard" condition; Calling MS DOS from Turbo Pascal; DOS 2.x sort and find filters; the SET Tools.

•Vol 3, No. 5: Windows done here; C programming books, Part 2; C code excerpts, Part 2; A menu-driven PC using DISPLAY.COM; Turbo Pascal and MASM: making them work together; dBASE II routines; Turbo Pascal 3.0; TopView Programmer's Toolkit.

•Vol. 3, No. 6: About C's HLL overhead; How to recover all your costs; Report on the Comdex and Usenix shows; The Santa Cruz Operation XENIX for the IBM PC; Common C bugs; CTOOLS2; Turbo Pascal 3.0 and subdirectories; More dBASE II routines; Clipper reviewed; A better way to interface BASIC and Assembler.

•Vol. 4, No. 1: FNE on PC/AT hard disk problems and plunging 64K DRAM prices; AUTOINC explained; Sorting using an AVL tree; Combining binary trees; Using DOS to call programs from Turbo Pascal; An operating system for the people; Final installment of Brett Salter's dTOOLS; Two microcomputer graphics books reviewed.

•Vol. 4, No. 2: FNE on Intel 80386, single- vs. multi-user, multi-tasking computing, benchmarks; The home office syndrome; Frank Greco's industry news; Funky string functions; Capturing file handles in Turbo Pascal; Proposed ANSI C Standard, Part 1; Optimizing for speed in Assembler; Memory resident control over print screen functions; Structuring BASIC output with field statements; QuickBASIC Compiler and networking; Programming for Printer Output reviewed.

•Vol. 4, No. 3: FNE—another RISC machine, the paging game, software viewpoint, programmer talent, and Aloha, 32032; Sun Microsystems' next generation PC; Consulting—reusable modules; INT24.PAS, a critical error handler for Turbo Pascal; Memory model independence in Assembler; An approach to windowing for the assembly language programmer; Proposed ANSI C Standard, Part 2; Microsoft and Lattice C 3.0 reviewed; Accelerator boards for software development reviewed.

•Vol. 4, No. 4: Hal W. Hardenbergh on adult toys, tomorrow's micros, HALGOL update, benchmarks, and RISCs; Frank D. Greco on parallel computers, RISC, UNIX; Consulting—efficient custom programming; DOS 2 file handles and file I/O; Programming the WordStar document file conversion utility; More optimizing for speed; Teach an old compiler a new trick; Memory resident programs in C; Tips on formatting with BASIC PRINT USING statements; The structure of dBASE II and III file headers.

•Vol. 4, No. 5: Hal W. Hardenbergh on 32-bit PCs, the 68040, Weitek's data-path part, etc.; Frank D. Greco on large screen monitors, language translators, UNIX, and more; Prolog intro, Turbo Prolog reviewed; Consulting—tax traps, consulting styles, nest eggs; ASM—Search and Display; ASM—the GROUP statement; The proposed ANSI C Standard, Part 3; C—measuring stack usage and Lattice C's startup code; Short boards reviewed.

•Vol. 4, No. 6: DTACK GROUNDED; Unix Books, Part 1; How many amoebas at one o'clock; DOS functions 45H and 46H; DOS Terminate and Stay Resident; Turbo Pascal inline code; Pascal—Assembler souce code for DOS function calls; Alignment problems in C and managing memory allocation; Proposed ANSI C Language Standard, Part 4; The Canon A1 laser printer; Genifer dBASE applications generator reviewed; Advanced Trace86 debugger reviewed.

•Vol. 5, No. 1: DTACK GROUNDED; The new DOS; Marketing and rules to live by; File redirection; Inside the EGA; String allocation in Pascal; ANSI C Language Standard, Part 5; More tips on formatting BASIC data.

•Vol. 5, No. 2: DTACK GROUNDED; Starting your own software house; Pink noise; TSR graphics print screen utility; Coding guidelines for resident programs; Parallel processing with the EGA; C portability in practice and using header files; Proposed ANSI C Language Standard, Part 6; Indexing in BASIC; A first look at Turbo C.

To order, just use the convenient form located on the perf insert, at left. If the card is missing, you may place credit card orders by calling our office at:
(503) 484-2162

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Formatting BASIC Output with Buffered PRINT USING

by

Paul Dymerski

An article by Bernard H. Robinson, Jr., in *PJ* Volume 4, Number 2 (March/April 1986), "Structuring Basic Output with Field Statements," presented the idea of using the BASIC FIELD statement to structure ASCII output to screen or printer. About the same time that this article appeared, I was trying to find a method that would allow me to format text and numbers similar to PRINT USING, but to a string in memory rather than to an output device such as a printer or disk file. Unfortunately, this capability is not present in Microsoft BASIC.

Reformatting data in memory has several advantages. In my program, for example, I wanted the user to have the option of directing output to several different devices. Rather than write separate PRINT USING routines for each device (assuming they may have different line widths and page lengths), it would be more efficient to write one routine that would format a line of output and then edit this formatted string for the particular device. Formatting to a string in memory would also be helpful for obtaining the exponent of a number by using the PRINT USING format of *##.###^^^^*, or for rounding numbers properly.

The BPU Method

Now, I'm sure you're expecting to read that I have written a 500 line assembly language program that mimics BASIC's PRINT USING function, that it is fast, and that it takes up only 2000 bytes of memory. I have such a utility but dread using it because it doesn't offer all of the formatting capabilities of good old PRINT USING. What I will present is fewer than a dozen lines of code written totally in Compiled BASIC that fool the PRINT USING function into believing that it is writing formatted strings to disk when it is not.

The example programs shown here are written for the Microsoft QuickBASIC compiler versions 1.0 or 2.0. While these routines could easily be rewritten for the BASIC Interpreter, the price of Microsoft's compiler is so cheap that it is hard to understand why anyone would want to use "SlowBASIC." The new compiler even includes a built-in editor at least as good as the Interpreter's version, and in memory compiling! Buffered PRINT USING (BPU) formatting is actually very straightforward and requires little knowledge of BASIC's confusing record functions such as FIELD, MKI$( ), MKS$( ), or MKD$( ). The PRINT USING statement takes care of all formatting for you! In BPU formatting, we will open a file buffer to temporarily store our formatted string, then recover it in another string without actually saving anything to the disk. As long as the file buffer is open, we can print one completely formatted string to it, or we can append pieces until our string is complete. Since no drive access has been made, our routine will be faster than a ram disk!

Example Program

Figure 1 is a sample program that outlines the BPU method as it would normally be coded in BASIC. If you are an experienced Compiled BASIC programmer, you may want to skip directly to Figure 2, which shows how the BPU method can be coded as a multi-line function in QuickBASIC. Since Compiled BASIC does not require line numbers, I have placed a "line label" at the start of each program section. A line label always ends with a colon.

Opening a Buffer

We can create a buffer to store our partially formatted string by using the normal OPEN statement to create a random access file on the default disk. We don't care what the name of this disk file is since we will erase it when we are finished. For our example I have named the file " - " (dash with no extension). The goal in buffered PRINT USING is to hold all our formatted characters in a file buffer in memory until we are ready to recover the completed string. DOS on the other hand, will try to move our characters from the buffer to a disk file as soon as possible to make room for more data. Once DOS moves our string, it will reset the buffer pointer to the first position so that any more characters entered will overwrite the previous characters. The disk access initiated by DOS will also slow down our routine, defeating our attempts to maintain a fast and efficient formatter. To cheat DOS we must be sure to set the size of our buffer larger than the largest expected string. In the example program in Figure 1, a buffer size of 256 characters has been created in z.open.buffer: and the field associated with the buffer is called z.mystr$.

Even though few if any of our strings will be exactly 256 characters long, a field length the size of the buffer will guarantee that no characters overflow the field and are lost. We will see how to handle shorter strings in a moment. For a random access file in QuickBASIC the default record length (that is, buffer length) is 128 bytes or characters. The maximum record length is 32,767 bytes.

Buffered PRINT USING

Z.create: demonstrates how PRINT USING can be used in the normal manner to format strings or numbers into the buffer. Since our output device is buffer number 253 in the example,

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

all formatted I/O must be directed to this buffer. PRINT USING statements can be as simple or as complex as BASIC allows. The example shows how two separate statements are used to append strings into the buffer. These could actually have been included in one statement. Notice that each PRINT USING statement is followed by a semicolon. A carriage return character, chr$(13), will be included in the output string if the semicolon is omitted. Likewise, even the final character appended to the buffer must not be followed by a carriage return.

Z.print.using: is the heart of the formatting routine. The buffer pointer is always sitting at the position just after the last character printed to the buffer. DOS knows where this pointer is located at all times, but this value is not easily accessible to us in BASIC, so we have no way of knowing how long our formatted string is. To terminate the string we can borrow a technique from assembly language programming called "null terminated strings." The only character never used in a string is the null character, chr$(0). If we append a null to the end of our formatted string sitting in the buffer, we can convert this string to a null terminated string. The end of our string is now the location of the first null character minus one (for the null character itself).

Now that we know the length of our string and the fact that it corresponds to the field mystr$, we can recover our formatted string into BASIC memory. In our first example this is returned in prntln$. The GET statement resets the buffer pointer to the first character of the buffer, in effect clearing the buffer for us. While the buffer actually contains leftover string fragments from previous formatting routines, our null terminated string will guarantee that we recover only the correct portion of the buffer containing our string.

The print.using$ Function

One of the features that makes QuickBASIC a "modern" compiler is that it permits multiple line sub-programs and functions. Figure 2 lists a self-contained version of the BPU method as a function called FNprint.using$. The features of this general purpose function include multiple buffers with simple control of buffer opening, closing, and size.

To use FNprint.using$, the function must be included in the QuickBASIC source code before it is called. Each time the function is called, the buffer channel number must be specified, which can range from 1 to 255. To open a buffer for formatting, the integer set% is set to the desired size of the buffer. If the buffer is opened correctly, set% is returned as zero. The function opens what should be a unique filename as the negative of the channel number. For channel 253, for example, the temporary file name would be "- 253" with no extension. The space that BASIC leaves between the minus sign and the channel number is removed to produce a legal filename. If by chance the file is not deleted before the program is terminated, the filename "- 253" with a space in the name cannot be erased with the DOS DELETE command. (The filename "- 253" could be deleted from a BASIC program).

Once the buffers are open, any legal PRINT USING statement can be directed to the buffer of choice. When a string is ready to be recovered, FNprint.using$ is accessed like a normal BASIC function with set% = 0. Since the buffer pointer is reset after each string recovery, the function can be called only once to recover the string. A second call will return a null string (""). The buffer, on the other hand, is now ready for another formatting sequence.

To close the buffer the function is called with set% = -1, or any negative number. In this case the function closes the buffer and deletes the empty disk file.

In QuickBASIC the DEF FN...END DEF compiler command allows multiple line functions to be defined. As in Interpreted BASIC, parameters can be passed to the function in parenthesis in the function name. In QuickBASIC, however, a function can have locally or globally defined variables and multiple definitions of the function itself. The value of the function is returned in the conventional manner.

To keep the FNprint.using$ function compatible with earlier versions of QuickBASIC, the routine does not use QuickBASIC's BLOCK IF statement. Instead the function is made into one large continuation line using the underscore and the IF...THEN...ELSE construct. The first IF statement tests to see if a formatted string return is requested. If so, the function converts the buffer string to a null terminated string, moves the buffer string to a memory string, resets the buffer pointer for the next formatting sequence, and exits.

If set% is greater than zero, the routine converts the channel number to a filename and opens a random file buffer of the length set by buf%. If set% does not meet either of these requirements (if it's negative), the function closes the current channel and deletes the file from the disk. Since file opening and closing needs to be done only once for all formatting, disk access is minimal.

Closing Comments

The BPU method may not be as fast as an assembly language program, but it is much smaller and guarantees BASIC com-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

TASKVIEW™

WHY GIVE UP. . .

BATCH FILES,

I/O REDIRECTION,

SIDEKICK,

DOS MENU PROGRAMS,

MOST OF YOUR RAM,

EXECUTION SPEED?

Compatible, efficient DOS multi-tasking.

We designed Taskview with efficiency in mind. During normal operation, TASKVIEW hides behind DOS, providing you with control of up to 10 concurrent or non-concurrent programs. Just the touch of a key instantly switches a program to the foreground. Included desktop utilities let you cut and paste from program to program. Simple to use and reasonably priced, no well equipped PC user should be without it.

Requires: PC/AT/Jr compatible, DOS 2.0-3.1, 256K RAM, 1 Floppy drive.

Taskview trademark of Sunnyhill Software
Sidekick registered trademark of Borland Intl.

30-day money back guarantee

Dealer Inquiries Invited.

$69.95 plus $5.00 S&H

Washington residents add 7.9% sales tax
International orders add $5.00
VISA and Mastercard accepted

To order Toll-Free
call 1-800-367-0651

Sunny Hill Software
13732 Midvale N. Ste. 206
Seattle, WA 98133
(206) 367-0650 M-F, 8-6 PDT

TURBO PROFESSIONAL™

SERVICE INTERRUPTS
No assembly required

RESIDENT PROGRAMS
Easy, pop-up routines

EXECUTIVE PROGRAMS
Run ANY DOS program

DISK SECTOR I/O
Lowest level access

FAST TEXT WINDOWS
Virtual windowing system

KEYBOARD MACROS
Simple, powerful

LOTS OF EXAMPLES
21+ full example programs

MUCH MORE. . .
Over 140 routines in all

"If you never thought Turbo Pascal was a systems programming language, you've never seen Turbo Professional."
Darryl Rubin
Computer Language

For programs that move with technology—Turbo Professional—a truly professional library of subroutines.

150 page reference manual.
Full source—many example programs.
No royalties charged for applications.

Requires IBM compatible, DOS version 2.0 or greater, Turbo Pascal 2.0 or greater.

Sunnyhill Software not affiliated with Borland International.
Turbo Professional, trademark of Sunnyhill Software
Turbo Pascal, registered trademark of Borland International

Dealer Inquiries Invited.

$69.95 plus $5.00 S&H

Washington residents add 7.9% sales tax
International orders add $5.00
VISA and Mastercard accepted

To order Toll-Free
call 1-800-367-0651

Sunny Hill Software
PO. Box 55278
Seattle, WA 98155
(206) 367-0650 M-F, 8-6 PDT

patibility with the PRINT USING function. For floppy disk systems, this method may be dangerous since any user may remove the floppy disk associated with the formatting buffers so that the files cannot be closed properly. Until Microsoft decides to build internal string formatting capabilities into BASIC, however, the BPU method can do wonders to improve the speed and versatility of many a program. □

Code follows.

Paul P. Dymerski has been working with Compiled BASIC for a number of years and now specializes in Compiled BASIC on micros. He is a major contributor to the Finally! family of libraries for Komputerwerk, Inc. A Ph.D in chemistry, he has authored over 35 articles in scientific journals, as well as many contributions to user group newsletters.

COPYRIGHTED © 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Code from Dymerski, Formatting BASIC

```
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'  Example Compiled Basic Program for Print.using to a string.
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

''''''''''' open a file buffer for formatting in ''''''''''''''''''''
z.open.buffer:
   open "-" for random as #253 len=256%  'use file buffer for formatting
   field #253, 256% as z.mystr$          'string returned as formatted

''''''''''' build string in buffer with print using ''''''''''''''''
z.create:
   print#253, using curfmt$;strval$;     'appending a formatted string
   print#253, using curfmt$;numval!;     'appending a formatted number

''''''''''' recover and print the formatted string ''''''''''''''''''
z.print:
   gosub z.print.using
   print prntln$

'''''''' Exit by closing print buffer and erasing file ''''''''''''''
z.exit:                                  'NORMAL exit here
   close 253                             'close the formatting buffer
   kill "-"                              'erase our dummy buffer
   end

''''''''''''' set end of string with null and recover ''''''''''''''''
z.print.using:
  print #253,chr$(0)                     'complete line with null
  i%=instr(z.mystr$,chr$(0))-1           'find last legal character
  prntln$=left$(z.mystr$,i%)             'recover formatted string
  get #253,1                             'reset buffer to beginning
  return
```

Figure 1. Compiled BASIC listing of Example BPU program

```
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'       TITLE: FNprint.using
' DESCRIPTION: REformat data to a string in memory.
'
'        CALL: x$ = FNprint.using(set%,buf%)
'       INPUT: buf%   Selects the file buffer channel (1 to 255)
'              set%   Controls the functions operation
'                     >0 Sets the maximum buffer size. This should be
'                        larger than the largest expected string length.
'                      0 Returns the formatted string accumulated in
'                        buffer buf% and resets for the next string.
'                     <0 Closes the buf% buffer and erases the empty file.
'      OUTPUT: x$     Formatted string created in buffer buf% when set%=0.
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

DEF FNprint.using$(z.set%,z.buf%)
  if z.set% = 0 then _                          'z.set control function
    print #z.buf%,chr$(0);_                     'complete line with null
    i%=instr(z.mystr$,chr$(0))-1:_              'find last legal character
    FNprint.using$=left$(z.mystr$,i%):_         'recover formatted string
    get #z.buf%,1:_                             'reset buffer to beginning
     else _
       if z.set% > 0 then _                     'use file buffer for formatting
         open "-"+mid$(str$(z.buf%),2) for random as #z.buf% len=z.set%+1:_
         field #z.buf%, z.set%+1 as z.mystr$:_  'str returned as formatted
         z.set%=0:_                             'open only once!
           else _
             close z.buf%:_
             kill "-"+mid$(str$(z.buf%),2)      'close buffer and remove
END DEF  'FNprint.using$

'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'          Example Compiled Basic Program for Print.using
'          to a string.
'                       by Paul P. Dymerski
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

' build string from pieces in buffer with print using

  buf1%=253                                 'buf1% selects file buffer number
  strfmt$="\      \= "
  numfmt$="#####.##"
  strval$="TEST NO."
  numval!=123.9857
  x$ = FNprint.using$(250,buf1%)            'open buffer length 250
                                           'accumulate a formatted string
  print#buf1%, using strfmt$;strval$;       'appending a formatted string
  print#buf1%, using numfmt$;numval!;       'appending a formatted number
  x$ = FNprint.using$(0,buf1%)              'return string and print to screen
  print x$

' create a formatted string all at once
' (leave buffer 253 open for now)

  buf2%=252                                 'open a second buffer
  strfmt$="\      \= #####.##"
  strval$="TEST NO."
  numval!=123.9857

   x$ = FNprint.using$(250,buf2%)                   'open buffer length 250
   print#buf2%, using strfmt$;strval$,numval!;      'format the line
   x$ = FNprint.using$(0,buf2%)                     'return string
   print x$
   x$ = FNprint.using$(-1,buf1%)                    'clear and close buffer #1
   x$ = FNprint.using$(-1,buf2%)                    'clear and close buffer #2

   print"It's that easy!"
 end
```

Figure 2. FNprint.using$ Function and example program

DAN BRICKLIN'S DEMO PROGRAM ONLY $74.95

Read what they're saying about this popular program for prototyping and demo-making:

"A winner right out of the starting gate. After you use DEMO once, you'll wonder how you got along without it."
—PC Magazine

"Everybody who writes software, either commercially or for in-house applications, should immediately order a copy. Period. No exceptions."
—Soft·letter

Product of the Month
—PC Tech Journal

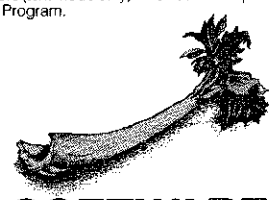
Thousands of developers and most of the largest and best known software companies are using this program. You can, too. Act now!

NEW TUTORIAL! JUST $49.95

The perfect companion to the Demo Program. The Tutorial helps you learn the ins and outs of its basic and advanced features. Complete with a 96 page manual containing step-by-step instructions, diskette, and function key template.

ORDER NOW!
1-800-CALL-800 x8088
VISA MasterCard American Express

Use 800-number for orders only.
Questions, special shipping, etc., call **617-332-2240.**
No Purchase Orders. Massachusetts residents add 5% sales tax. Outside of the U.S.A., add $15.00.
Requires 256K IBM PC/Compatible, DOS 2.0 or later. Supports Monochrome, Color Graphics, and EGA Adapters (text mode only). The Tutorial requires the Demo Program.



SOFTWARE GARDEN, INC.
Dept. J
P.O. Box 373, Newton Highlands, MA 02161

Number **108** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

2 Programmers & Developers New Products!

Distribute Your Demos with No Royalties

Screen Machine creates interactive demos, tutorials, menu systems and DOS shells. Includes a text screen editor that optionally generates source code* and binary or text files. Never write code for screen display again. Capture any program's text screens for editing and your own use. Capture CGA compatible graphics screens for BLOAD or direct display. SAVE hundreds of HOURS of work.

Now there's no need for separate screen and demo software packages and no need to pay outrageous royalties. Priced at only $79.00.

*Turbo Pascal, Mach 2 for Turbo, Assembler, dBASE II & III, BASIC (including The Inside Track and Mach 2).

Supercharge Turbo Pascal

Mach 2 for Turbo Pascal adds assembler speed to your programs. 90 + subroutines, most in assembler, give you speed and functionality you never knew was possible. No knowledge of assembler language required.

INSTANT displays. INSTANT windows (incl. exploding and boxed). FASTEST sort you've seen. Read/write files FAST as DOS. INSTANT menus, 1-2-3 horizontal and vertical bar.

Trap ^C/^Break & DOS critical errors so no more A)bort, R)etry or I)gnore. Emulate BASIC PRINT USING for FAST formatted numbers. Execute any prog, batch or DOS command without ending program.

Read environment. Read file directory. Get/set file attributes. Plus too many string functions to describe here. No royalties when you distribute COM programs. All source code included. A true bargain at $69.00.

NOT COPY PROTECTED. 30 Day Money-Back Performance Guarantee. Requires IBM/compatible & DOS 2 +.

Order Now 800-922-3383

We welcome VISA/MC. COD US only $3. S/H US $3, Canada $5, Elsewhere $18. GA res. add tax and call 404-973-9272. Demo available. Send $5 check. Refunded on direct purchase.

We also publish Stay-Res, Mach 2 for BASIC, The Inside Track and Peeks 'n Pokes.

MicroHelp, Inc.
2220 Carlyle Drive
Marietta GA 30062

Number 90 on the Reader Service Card

386 Products

Jet 386

Orchid Technologies
45365 Northport Loop West
Fremont, CA 94538
415/490-8586
$1199

The Jet 386 is Orchid Technologies's 16MHz 386 accelerator card for the AT. It takes up a 16-bit slot, is cabled to the host motherboard at the host CPU socket, and moves the host 286 onto the card. It uses the AT's memory, with the assistance of an on-board 64K, 32-bit, fast, no-wait-state cache.

In that it has a cache, the board is built along the lines of Orchid's 286 TinyTurbo accelerator rather than their 286e, which is a co-processor board with a full 1MB of memory. This reduced on-board memory is probably one factor in the Jet's surprisingly low price. Performance is correspondingly decreased.

Performance for the Jet 386 is claimed at 2–5 times an AT versus 3 times an AT for Orchid's 12MHz 286e co-processor board (which retails for about $1500).

If you have an AT with a "speed cop" that limits the boot to 6MHz, a way is provided for booting from the 286 and then switching to the 386.

The 64K on-board 32-bit RAM holds the BIOS and caches the first 4MB of AT RAM. A connector is included for a future 32-bit RAM expansion daughtercard of up to 16MB.

The board has a socket for a 387 math chip, but it will run the 287. An 8MHz or 12MHz 80287 math chip adapter for the board is another $95. (The 287 math chips themselves are available from Orchid for $375 and $550, respectively.)

Orchid provides this overview of the 386 accelerator market:

Company	Product	Price	RAM
Orchid	Jet 386	$1199	64K cache
Sigma Designs	CPU 386	$1600	
	Turbocache 386	$1600	32K or 128K
Intel	Inboard 386	$1995	64K cache
American	386 Turbo	$1995	64K, 1M
Applied Reasoning	PC-Elevator 386	$1995	2M
Seattle Telecom	386 Board	$3495	2M
Goldhill	Hummingboard	$7000	6M

One final note: Have you noticed that computer companies have been straining to wring more and more out of the word "Turbo"? Now Orchid has revved up the word "hyperspeed" for the Jet 386. Have we approached that unassailable limit, the speed of hype? A friend has even pointed out an advertisement for "Turbo Dahlias" in her plant catalog. For God's sake, let's back off before the wings come off.

386 Motherboard

Zeos International
530 5th Avenue NW
Suite 1000
St Paul, MN 55112
800/423-5891
$1995

Here's a 10-layer, AT-compatible 386 motherboard that will replace the motherboard in a PC, XT, or AT. The board features an 8MHz, 16-bit AT-compatible peripheral bus and a 32-bit, 16MHz, zero-wait-state bus for memory. The board comes with no memory. It uses Chips and Technologies's seven chip AT/386 VLSI chip set and C&T's new 82C206 integrated peripheral controller. Sockets are also available on the motherboard for C&T's two-chip EGA, Western Digital's single chip floppy controller, and two serial ports and one parallel port. The board's total chip count is under 60.

The Phoenix 386 AT-compatible BIOS with EGA extension is included for compatibility with DOS through 3.2.

The board requires a minimum of 1 MB of system memory and has space for up to 16MB in 256K or 1MB SIMM RAM modules. Zeos says that the board will run with a 20MHz 386 when they become available. An 80387 math co-processor socket is provided although the chip, of course, is not yet available.

The PC version has four 16-bit slots, the XT version three 16-bit and two 8-bit, and the AT version, four 16-bit and three 32-bit slots.

PRODUCED 2004 BY SINZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Intel versus TI

The premiere issue of a Microsoft house publication, the *Systems Journal*, has articles on Intel's and TI's graphics chips.

Intel 82786

The Intel chip is touted as drawing graphic primitives at 2 million pixels per second and creating text at 25,000 characters per second. Faster than you can read. A single 16-bit request from the CPU will create points, lines, rectangles, circles, polygons, and other primitives.

The graphics instruction set looks like this:

Bit-Block Transfer within Bit-Map
Bit-Block Transfer across Bit-Maps
Define Texture: Opaque
Define Texture: Transparent
Define Character Set: Word
Define Character Set: Byte
Define Bit-Map
Define Colors
Define Logical Combination
Define Clip Rectangle
Define Intercharacter Space
Define Character Orientation
Draw Arc: Exclusion
Draw Arc: Inclusion
Draw Point
Draw Incremental Points
Draw Series of Horizontal Lines
Draw Line
Draw Rectangle
Draw Polygon
Draw Polyline
Draw Circle
Draw Character String: Opaque
Draw Character String: Transparent
Enter Pick Mode
Exit Pick Mode
Move to Absolute Position
Move to Relative Position

Microsoft quotes Richard Bader at Intel as saying that when an 82786 board-level product comes to market, it will have a Windows driver. Intel and Microsoft have a cooperative program to provide such a driver. Bader also mentions that the chip will support large-format screens and full-page displays, with a resolution in the order of 1200x800, 1400x1400, or 1000x800 with gray scales.

TI TMS34010

The TMS34010 is a 32-bit graphics processor that runs at 6 MIPS. TI calls it a "RISC-like" chip—perhaps the new safe term, given the uncertainty about RISC architecture. Anyway, it executes most of its instructions in a single 160ns cycle. The chip has thirty 32-bit general-purpose registers, a barrel shifter that can rotate a field of up to 32 bits in a single instruction cycle, and a 256 byte instruction cache.

The graphics instruction set:

Add Registers in XY Mode
Compare X and Y Halves of Registers
Compare Point to Window
Convert XY Address to Linear Address
Draw and Advance
Fill Array with Processed Pixels: Linear
Fill Array with Processed Pixels: XY
Pixel Block Transfer: Binary to Linear
Pixel Block Transfer: Binary to XY
Pixel Block Transfer: Linear to Linear
Pixel Block Transfer: Linear to XY
Pixel Block Transfer: XY to Linear
Pixel Block Transfer: XY to XY
Pixel Transfer: Register to Indirect
Pixel Transfer: Register to Indirect XY
Pixel Transfer: Indirect to Register
Pixel Transfer: Indirect to Register
Pixel Transfer: Indirect XY to Register
Pixel Transfer: Indirect XY to Indirect XY
Subtract Registers in XY Mode
Line Draw

Kevin McDonough at Texas Instruments says that a distinguishing feature of the chip is that it is not limited to certain algorithms. Rather, any algorithm, be it line, ellipse, or circle, can be programmed, and the chip can be programmed for any format of text.

Texas Instruments and Microsoft also have a cooperative program to provide Windows drivers for systems designs using the TMS34010.

Rendition I

Renaissance GRX
Bellevue, WA
206/454-8086
$535

Renaissance has introduced a graphics controller for the PC based on the TI 34010. It provides full emulation of MDA, CGA, and Hercules displays, with EGA emulation as an option, and provides graphic resolution up to 1024x768 pixels. The company claims that graphics will run up to 100 times faster than on an IBM EGA. Renaissance has designed and developed its own Windows driver, as well as drivers for 1-2-3, Symphony, and AutoCAD. Renaissance will market the controller direct to OEMs for $535 each in sample quantities as of April, '87. The EGA add-on costs an additional $123.

Let's C 4.0

Mark Williams Company
1430 West Wrightwood Avenue
Chicago, IL 60614
$75

Version 4.0 features in-memory compilation, an integrated "MicroEMACS" editor that automatically locates source code errors dur-

MULTITASKING

Introducing

MultiDos Plus

The new multitasking software for the IBM-PC.

Ideal for developing applications in process control, data acquisition, communications, and other areas. Check these features which make **MultiDos Plus** an unbeatable value.

- Run up to 32 programs concurrently.
- Your software continues to run under DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in any language.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/resume programs.
- Programmatic interface via INT 15H for the following.
  - * Intertask message communication. Send/receive/check message present on 64 message queues.
  - * Task control by means of semaphores. Get/release/check semaphores.
  - * Change priority-128 priority levels.
  - * Suspend task for specified interval.
  - * Spawn and terminate external and internal tasks.
  - * Disable/enable multitasking.
  - * and more!
- Independent foreground/background displays.
- Access to DOS while applications are running.

Hardware/Software Requirements

IBM PC/XT/AT or true clone. Monochrome/CGA display adaptors or equivalent cards only. Enough memory to hold **MultiDos Plus** (48 KB) and all your application programs. Also may need 4 or 16 KB memory for "hidden screens" for each active task. MS-DOS (or PC-DOS) 2.0 or later operating system.

ONLY $49.95

Outside USA add $5.00 shipping and handling.
Visa and Mastercard orders call toll-free: 1-800-367-6707. In Mass call 617-651-0091, or send check or money order to:

NANOSOFT
13 Westfield Rd, Natick, MA 01760

MA orders add 5% sales tax. Write for source code and quantity price.

Number **109** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DeSmet C
SUPPORTS
ROMABLE CODE

DeSmet C ***$109**
Data & Stack in RAM
Code executes out of ROM
Full K&R + V7 extensions
Inline asm
Assembler, linker, librarian
Full screen editor (SEE[tm]),
8087 & S/W floating point

DeSmet C **$159**
with D88 Debugger
Set Breakpoints by line number or function name
Examine Global/Local variables by name
Show C source while debugging
Both D88 and User displays

Large Memory **$209**
DeSmet C w/ D88
32-Bit Pointers (Full Megabyte Addressing)
Fast Access of Static Data
Includes the standard compiler features mentioned above.

*Both D88 & Large Memory available as options at $50 each.
We also have a C compiler for the Mac. $150. Call for details.

— plus —

Graphics **$35**
BASICA-like (+src)

Hacker **$25**
Source for start-up, RAM.COM & more

Make **$50**
Full UNIX-level

Tools (+source) **$35**

XARRAY **$39**
Large Arrays (+source)

C WARE
CORPORATION
P.O. Box C
SUNNYVALE, CA 94087 USA
(408) 720-9696 TELEX: 358185
Street Address: 505 W. Olive, #767
VISA, MC & AMEX accepted

Number 38 on the Reader Service Card

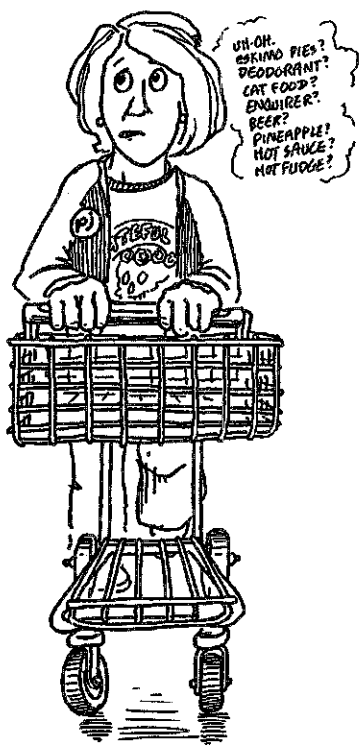
ing compilation, and development features such as 80286 code generation, ROMable code, 8087 math co-processor support, and large and small memory models. Four Unix-compatible file processing utilities are also included. Mark Williams's fine large and small model source-level debugger, CSD, is available as an option.

The package is full K&R, with ANSI extensions for void, enum, and structure assignments.

Robin Swartz, president of Mark Williams, says that in Version 4.0, compile-edit time has been halved by using an integrated editor/compiler and by RAM-based compilation.

CONTROL/386 for 80386 Computers
Phoenix Technologies, Ltd.
617/769-7020

Here from Phoenix is a systems software product that is intended to smooth the way for software and hardware peripherals running on a 386 machine. Control/386 compensates for slower physical devices by, for example, caching disk drives and by copying code from slower ROM devices into 32-bit extended memory RAM. It also provides 32-bit emulation of the EEMS and EMS specifications, emulates some undocumented 80286 instructions not present in the 386, and emulates the 8088's 1MB memory limit. It also runs timing-dependent copy-protection schemes successfully on the 386. □



RANDOM ACCESS MEMORY AGAIN.

PORTABLE POWER WITH MasterFORTH

Whether you program on the **Macintosh**, the **IBM PC**, an **Apple II** series, a **CP/M** system, or the **Commodore 64**, your program will run unchanged on all the rest.

If you write for yourself, MasterFORTH will protect your investment. If you write for others, it will expand your marketplace.

Forth is interactive – you have immediate feedback as you program, every step of the way. Forth is fast, too, and you can use its built-in assembler to make it even faster. MasterFORTH's relocatable utilities and headerless code let you pack a lot more program into your memory. The resident debugger lets you decompile, breakpoint and trace your way through most programming problems. A string package, file interface and full screen editor are all standard features. And the optional target compiler lets you optimize your application for virtually any programming environment.

The package exactly matches *Mastering Forth* (Brady, 1984) and meets all provisions of the Forth-83 Standard.

MasterFORTH standard package	$125
(Commodore 64 with graphics)	$100
Extensions	
Floating Point	$60
Graphics (selected systems)	$60
Module relocator (with utility sources)	$60
TAGS (Target Applic. Generation System) – MasterFORTH, target compiler and relocator	$495
Publications & Application Models	
Printed source listings (each)	$35
Forth-83 International Standard	$15
Model Library, Volumes 1-3 (each)	$40

(213) 821-4340

MICROMOTION
8726 S. Sepulveda Bl., #A171
Los Angeles, CA 90045

Number 13 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

HELP

is at hand

HELP/Control™ — an on-line help system for the IBM-PC. **HELP/Control** includes **HELP/Runtime, HELP/Popup** and our help screen compiler.

With **HELP/Runtime**, a few simple subroutine calls add context sensitive on-line help to your application. **HELP/Runtime** includes tested interfaces for C (Microsoft and Lattice), Pascal (Microsoft and Turbo), IBM BASIC (Interpreter and Compiler), Microsoft FORTRAN, COBOL (IBM and Realia) and assembler.

Use our concise screen definition language to build your help files. You define the bold captions on your help screens and specify the links to other screens. If you have existing documentation files, we supply a program which automatically marks them up to get on-line quickly. You can put an entire user or reference manual on-line, completely accessible to the user at all times.

HELP/Control also includes **HELP/Popup**, which provides memory resident access to your custom help screens. Use it to document dBase and 123 applications. **HELP/Popup** uses the same help files as **HELP/Runtime** and operates the same from the user's point of view, allowing you to provide a consistent on-line environment across diverse applications.

The complete package (software, on-line manual, printed manual, and demo programs) costs $125.00 and includes a royalty-free license to add **HELP/Runtime** to your applications and a license to make 25 copies of **HELP/Popup**. A demonstration diskette, including the on-line manual, costs $15.00. To order, or for more information (including dealer, multiple-copy and site-license pricing) call MDS at 207/772-5436. We accept MasterCard and VISA.

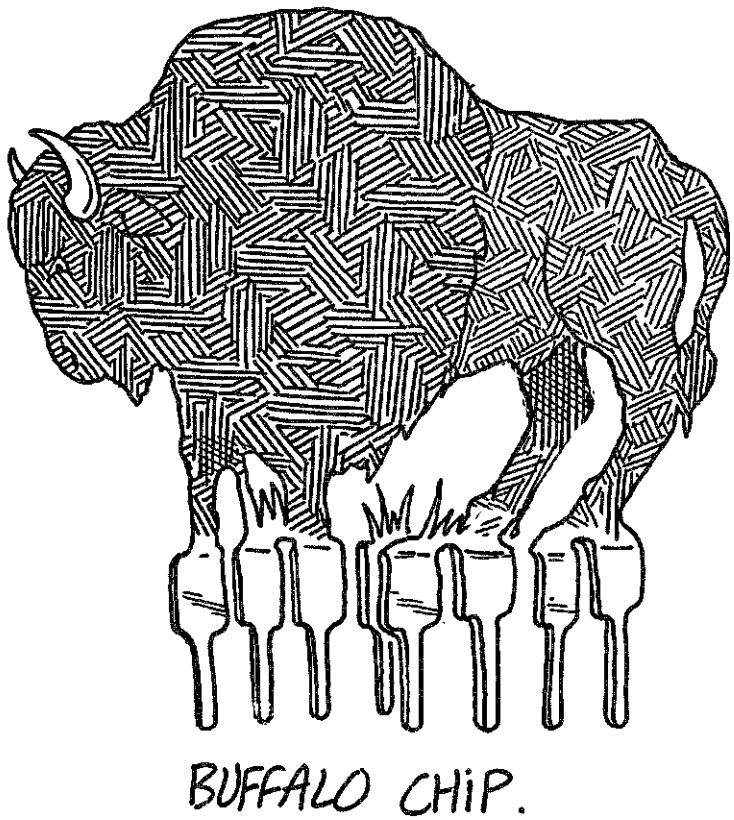
New options with release 1.20:

- **Use Dan Bricklin's Demo Program to build your help system. Our translator turns it into a HELP/Control source file.**
- **Include graphics in your help screens.**
- **Mouse support.**

MDS

MDS, INC., P.O. BOX 1237, PORTLAND, MAINE 04104

Number **69** on the Reader Service Card



New Compiler Language

PL/D *is FAST*

...and comes in source

System Oriented

PL/D™ is used to write utilities, operating systems, systems software, and to access all memory, registers and I/O.

Fast Compiles

PL/D compile rate of 6000+ lines per minute (LPM) is the PC's fastest:

Device	Time (sec)	Rate (LPM)
Floppy	23.7	6759
Hard Disk	17.7	9051
Ram Disk	13.7	11593

All times 4.77 MHz 8088 for 2670 lines.

Fast Run Time Code

PL/D is written entirely in its own language and shows the speed of its generated code. FOR-NEXT is tuned for maximum loop speed. Subroutine parameters pass in registers.

Unifies Program

PL/D incorporates assembly level coding using the same syntax, listing format, and naming rules as high level parts of the program. There is no source or run time penalty to intermix high and low level code.

Unifies Debug

PL/D lists code, variable and subroutine call locations on same line as source image. Addresses usable in debug session directly; no translation required.

Extends Via Macros

PL/D allows you to define MACROs for extending the language. You can build tree, linked list, and ordinary linear MACROs. Control provided by directives:

IF#-THEN#-ELSE#-ENDIF#
CHOOSE#-CASE#-BUT#-ENDCH#
WHILE#-ENDWH#
FOR#-NEXT#

Extends Via Source

PL/D source allows you to learn compiler internals and extend for your own needs.

Demo **$4.95**
Limited compiler, user manual on disk.

Object **$49.95**
Full compiler, printed user manual.

Compiler **$124.95**
Full compiler and source. Printed user and internals manuals.

Not copy Protected. Requires PC, XT, AT or compatible with 2.0 MSDOS or above and 192K. Distributed on 5 1/4 inch floppy diskettes. Add tax in CA. 30 day guarantee.

DAIR Computer Systems
3440 Kenneth Drive
Palo Alto, CA 94303
(415) 494-7081

Number **5** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Instant Replay™
Version II

Demonstrations
Presentations
Prototypes
Tutorials
Menus
Music

"Indispensable . . . A clear improvement over Dan Bricklin's Demo Program."
PC Magazine

"Incredible . . . We built our entire Comdex Presentation with Instant Replay."
Panasonic

"Instant Replay brings new flexibility to prototypes, tutorials, & their eventual implementation."
Electronic Design

"I highly recommend Instant Replay."
Computer Language

Build tutorial replays of actual programs, or animated prototypes. Memorize and replay keystrokes and pause times. Has the unusual ability to insert prompts, pop-ups, prototypes, user involvement, music, and branching menus into replays.

Includes Screen Maker, Keystroke Time Editor, Prototyper, Text Editor, Music Maker, Menu Maker, Screen Grabber, Animator, Control and Insertion Guides, 250 Page Manual, Online Tutorial. (Not Copy Protected, No Royalties). 60 Day Satisfaction Money Back Guarantee.

Instant Replay II ***$149.95***
Demo Diskette $5 *Dealer Poster $3*

Call or Write. We accept Visa, AMEX, Master Card, COD, PO. Dealer Inquiries Welcome.
Outside US pays postage.

Nostradamus
3191 South Valley Street (ste 252)
Salt Lake City, Utah 84109
(801) 487-9662

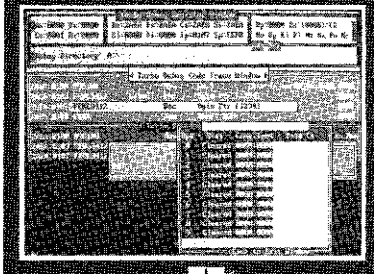
SPEEDWARE™
A DIVISION OF AMERICAN AFFORDABLE SOFTWARE CORPORATION

PRESENTS

Turbo Debugger™ Features
- User-definable custom windows — Change window sizes or colors
- Single-step tracing with "Back-Tracing" up to 1Kb instructions
- In-Line relocating Assembler with non-destructive insert capability
- Fully Symbolic, Create Comments and Line Labels, Keyboard macros
- Dual Monitor support, with Screensave
- Specify several Break-point conditions with the built-in Assembler
- Use NMI Breakout switch or User-definable hotkeys
- Read, Write and Modify EXE files

Turbo Editasm™ Features
- Integrated WordStar compatible Editor
- Integrated MicroSoft compatible Macro Assembler
- Full 8088, 8086, 8087, 80130, 80287, 80286(R), & EPI support
- Full Macro expansion capability
- Built-in Calculator & Xreference Capability
- Limited DOS command support
- Supports In-Memory Assembly

To maximize your programming potential order the ultimate "Turbo Team" today — the first well-tuned, high-speed assembler/debugger for the IBM P.C. and compatibles family.



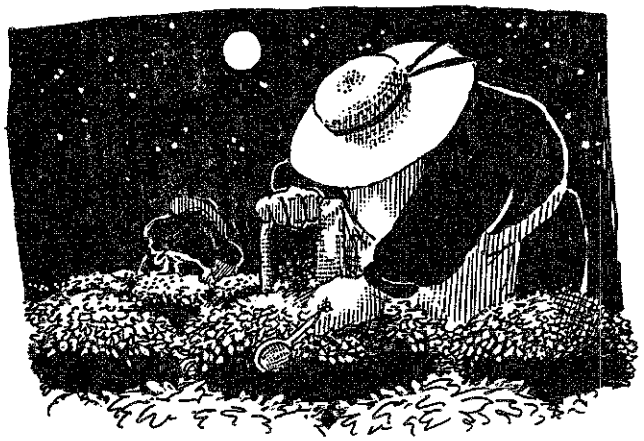
$99.00 Turbo Editasm™
$89.00 Turbo Debugger™

THE TURBO TEAM™
Inside California (800) 334-3664 • Outside California (800) 443-6190
$99.00 ☐ Turbo Editasm™
$89.00 ☐ Turbo Debugger™
Name
Shipping Address
City
State Zip
Telephone
Amount Enclosed
Payment VISA MC BANK DRAFT CHECK
Credit Card Exp. Date
Card #
CA Residents add 6% sales tax
Outside USA add 10% and make payment by bank draft payable in U.S. dollars drawn on a U.S. bank.

Speedware™
9719 Lincoln Village Dr. Ste. 303
Sacramento, CA 95827

Dealer and Education Discounts Available.

Number 72 on the Reader Service Card



LATE HOURS DURING THE BIT HARVEST IN THE FERTILE WILLAMETTE VALLEY.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Complete Accounting Software with Source Code

Create Custom Vertical Accounting Modules

• General Ledger • Inventory
• Accounts Payable• Cost Accounting
• Accounts Receivable

Buy all five software modules for $200 or purchase them separately for $50 each. Each module comes complete with full documentation, an executable file and documented pretty-printed source code.The modules share an easy-to-use common user interface and work together to make accounting tasks simple. A low-cost OEM license is now available for software developers.

System requirements: Turbo® Pascal 3.0 or above, the Turbo Pascal Toolbox, and an IBM® PC or compatible. Each module is shipped on a DSDD 5 1/4" diskette. Shipping charges: $5 US, $10 Canada. Send check or money order to:

PS BUSINESS SYSTEMS

PS Business Systems
4400 Wallingford Ave N.
Suite 3
Seattle, WA 98103
(206) 632-4568

Number **82** on the Reader Service Card

$99 **BUSINESS BOARD**

24 hour business information center using modems at 300/1200/2400 baud. Has a multi-level menu system that is easily customized with no programming req'd. Includes remote PC operations and integrated data base management. Source code additional charge.

$99 PC **COMMX** $119 CP/M

Emulates VT100,Wyse,HP,ADM,TV,IBM,ADDS,file transfers: KERMIT, XMODEM, COMMX mainframe, Telex/TWX. Instant DOS/foreground switch! Unattended macro controls and 700 entry dial directory. Electronic mail sub-system!

$59 **C DATA ENCRYPTION**

Data Encryption Standard (U.S. government standard FIPS PUB46) in Microsoft "C". Includes compression & telecomm formatting, allowing for faster transmission & storage on any computer or service. Complete "C" source code provided for additional $249.

HAWKEYE GRAFIX Inc
Box 1400, Oldsmar
Florida 33557
Call 813-786-8161

Number **57** on the Reader Service Card

WINDOWS—MENUS—DATA ENTRY—SCREENS

HI-SCREEN XL™

Application Developers! **HI-SCREEN** *XL*™, the newest version of HIGH SCREEN, is a complete and unmatched programming tool for managing your user-interface. **HI-SCREEN** *XL*™ features a revamped and enhanced screen editor, increased speed and performance, along with a new thorough reference and user's manual.

Whatever tool you own: compare, you'll be convinced!

1. **WINDOWS**
HI-SCREEN *XL*™ makes you a master in window management:
- Pop-up windows any size, anywhere.
- Use windows for data entry, menus, on-line help.
- Create context-sensitive help, conditioned by user keystroke sequence, with scrollable text.
- Unlimited number of windows, overlapping up to 26 levels deep. Save/restore area under each window.

1. **WINDOWS:** for on-line help, menus, data entry. Up to 26 levels deep.
2. **MENUS:** pop-up, pull-down or Lotus-style, also for batch files.
3. **DATA ENTRY:** automatic field checking: format, type, range... field by field or full screen. Error handling.
4. **SCREENS:** full-featured screen editor. Language independent. Fast display from RAM.

Royalty free, not copy protected, 30 day money back guarantee, trade up available.

Software, Inc.
PC|SOFT Product Line
500 Sutter St., Suite 222
San Francisco, CA 94102

(415) 397-4686
HI-SCREEN *XL*™ is $149
(CA res. add tax), S&H USA $5
Visa, M/C welcome

PASCAL, C, dBASE, BASIC, COBOL, FORTRAN, ASSEMBLER, etc.

Number **110** on the Reader Service Card

MEMORY RESIDENCY MADE EASY

Jack™ The Resident Program Developer's Kit, contains everything you need to create your own RAM resident software without the headaches of memory management, windows or DOS re-entrancy. What makes Jack™ unique is NO hooks and NO special code, just link Jack™ with your program. That's all there is to it. Now you can concentrate on how your program should run instead of worrying about how to make it memory resident.

Jack™ FEATURES:
- Virtually any C program can be made memory resident without any modification to the existing source code. Screens are automatically saved to conventional memory [or the expanded memory of an Intel Above Board] when popping up and restored upon popping down.
- DOS function calls and disk I/O can safely be used. Jack™ does not use any undocumented DOS calls to solve the DOS re-entrancy problem. [DOS 2.0 or greater required]
- Peaceful coexistence with other popular pop-up utilities.
- An easy to use windowing system, with source code, to minimize development time while maximizing flexibility.

Finally, a standard for easily creating memory resident software has arrived!

To Order: Please remit $199.95 [US] + S/H for Jack™ by VISA, written P.O., certified check or international money order to:

CrackerJack
Microsoftware Corporation.
200 Bay Street PO Box 86,
Toronto, Ontario. M5J-2J2. Canada.
(416) 865-9621.
Please specify either the Datalight C 2, Lattice C 3.0 and 3.1, or MicroSoft 4.0 version of Jack™.

Attention Software Developers, let's work together, please contact us regarding our distribution agreement.

Jack™ and CRACKERJack are registered trademarks of CrackerJack Microsoftware Corporation, Datalight is a registered trademark of Datalight Inc., Lattice is a registered trademark of Lattice Inc., Intel and ABOVE BOARD are registered trademarks of Intel Corporation, Microsoft and DOS are registered trademarks of Microsoft Corporation.

Number **88** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ BILLBOARD

PJ Billboard *is a directory of products and services of interest to the IBM PC programmer. The goal is to provide our readers with a wide range of information on programming products, introductions, and updates.*

PJ Billboard *serves advertisers by providing a low cost, flexible format in which they can present their products to PJ readers. The standard layout is four columns, 9 picas wide, with copy set in 7 point on 9 point leading. A* ***PJ Billboard*** *listing offers the advertiser a 400-character product description including company name, address, and telephone number. The listings are $89, prepaid, non-commissionable, for a one-time insertion. Listings must be received typed or legibly printed, with boldface or italic words indicated where desired. A convenient order form is available upon request. Copy deadlines are the same as the normal advertising space reservation dates, which can be found on our rate card.*

For more information call the
Programmer's Journal
Advertising Department
(503) 484-2162

FREE PUBLIC DOMAIN SOFTWARE FLYER!

IBM & Compatibles (MS-DOS) More than 4000 programs on 165 disks for only $3.50 per disk! Games, Educational, Business, Personal, Utility and Miscellaneous Software! Terrific software and service at unbelievable prices!

Send for FREE FLYER to:

AP-JP Enterprises, Inc.
P.O. Box 1155
W. Babylon, NY 11704

Tools for serious C programmers

access systems capabilities, build & maintain windows and menus for an application or to share between applications, create & maintain indexes — No royalties
Send for FREE FLYER to:

FLEMING SOFTWARE
P.O. Box 21106
Columbus, OH 43221-0106
(614) 267-2143

Pocket References for C Programmers and UNIX and DOS Users

C Library Reference — $6.00
MS-DOS Reference — $3.00
UNIX Command Summary —$6.00
Exploring UNIX Environment — $16.95
(Specify System V, BSD 4.2/4.3, or XENIX V)

Other references, UNIX software, and consulting available.

To order or send for our free catalog write or call:

SSC, Inc.
P. O. Box 55549-P
Seattle, WA 98155
(206) 367-UNIX

DOS USER'S GROUP

QUARTERLY NEWSLETTER
Latest MS and PC DOS news
New product reviews
Discounts on third party products

Third Party Demo Diskettes

Annual Membership fee U.S. $25.00
(Canada-Europe) $35.00

DOS USER'S GROUP
P. O. Box 26601
Los Vegas, NV 89126

EGA Card — $199.00
EGA Monitor — $340.00
XTerminal Modem 2400 Baud —$235.00
AT System (286) — $995.00
AT 386 System — Call $$$

Western Data Products
155 Corliss Ln., Suite 8
Eugene, OR 97404
(503) 689-3353

ADVERTISER INDEX

Reader Service Number	Advertiser Name	Page Number
53	Allen, Emerson & Franklin	41
86	Aspen Systems	29
30	Austin Code Works	49
36	B C Associates	5
2	Blaise Computing	3
3	Borland International	C4
38	C Ware	60
107	C. Abaci	49
64	Chalcedony Software.	19
56	CompuView Products, Inc.	9
88	Crackerjack Software	63
*	Creative Programming Consultants	48
83	Crescent Software	28
5	Dair Computer Systems	61
103	Desktop Ai	21
106	Dr. Dobb's C Tools	43
111	ESP Software	8
8	Ecosoft, Inc.	1
11	Gimpel Software	28
57	Hawkeye Grafix	63
91	King Computer Services	27
59	L.U.C.I.D.	49
63	Laboratory Microsystems, Inc.	34
44	Lahey Computer Systems	45
92	Manx Software Systems	15
69	MDS, Inc.	61
45	MetaWare	42
90	MicroHelp	58
13	Micromotion	60
105	Micro/Systems Journal	37
104	Microway	23
94	MMC AD Systems	42
95	Mystic Canyon Software	41
109	NanoSoft	59
112	Nostradamus	62
16	Opt-Tech Data Processing	29
102	Oregon Software	C3
26	Periscope Company, The	22
96	Phar Lap Software	29
26	Polytron	52
*	Programmer's Journal	53
82	PS Business Systems	63
*	Quaid Software	33
46	Quilt Computing	49
20	Rational Systems	26
93	Software Connection	42
98	Software Family, The	7
108	Software Garden, Inc.	57
110	Softway	63
72	Speedware	62
85	Sunny Hill Software	56
73	Whitewater Group	C2

* These Advertisers would prefer to be contacted directly

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Pascal-2

Digital's VAX has become the foundation for a community of development systems – PDP-11s, 68000s and PC-ATs – yet, until now, no high-performance development language spanned the range of architectures and operating systems. Now there is one. **Pascal-2**

Your VAX can become the development vehicle for PDP-11 code (RSX or RT-11 target), for MS-DOS applications, or for 68000/20 VERSAdos and UNIX systems through **Pascal-2** cross-compilers. And you can even turn the solution around and use our MS-DOS-to-VMS cross-compiler to offload your VAX. (We know what happens to VMS system response when six developers compile programs simultaneously!)

Oregon Software's matrix of **Pascal-2 native and cross-compilers** creates a single high-level development environment to solve the programming problems faced by the world's toughest industries – aerospace, communications, robotics, process control, medicine.

Pascal-2 is ISO standard at Level-1, the world's only certified Pascal Native and cross-compiler system. You get the portability, readability and reliability of a high-level language and structured access to the operating system or hardware when you need it. All in a rugged compiler that delivers the smallest, fastest code available.

Dramatically improve your productivity and introduce your VAX software to the PC next door.

Call or write OREGON SOFTWARE, INC. 6915 SW Macadam Avenue. Portland, OR 97219
(800) 367-2202 TWX: 910-464-4779 FAX: (503) 245-8449

OREGON SOFTWARE

Real tools for real work

AT LAST THE PERFORMANCE IS PORTABLE

The following are trademarks: Oregon Software, Pascal-2, Oregon Software, Inc.; IBM, PC-AT, PC-DOS, International Business Machines Corporation; MS, Microsoft Corp.; VERSAdos, Motorola, Inc.; PDP, RSX, RT-11, VAX, Digital Equipment Corp.

Number 102 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

NEW

Turbo C®

Turbo C: The fastest, most efficient and easy-to-use C compiler at any price

Compilation speed is more than 7000 lines a minute, which makes anything less than Turbo C an exercise in slow motion. Expect what only Borland delivers: Quality, Speed, Power and Price.

Turbo C: The C compiler for amateurs and professionals

If you're just beginning and you've "kinda wanted to learn C," now's your chance to do it the easy way. Like Turbo Pascal, Turbo C's got everything to get you going.

If you're already programming in C, switching to Turbo C will considerably increase your productivity and help make your programs both smaller and faster. Actually, writing in Turbo C is a highly productive and effective method—and we speak from experience. Eureka: The Solver™ and our new generation of software have been developed using Turbo C.

Turbo C: a complete interactive development environment

Free MicroCalc spreadsheet with source code

Like Turbo Pascal® and Turbo Prolog,™ Turbo C comes with an interactive editor that will show you syntax errors right in your source code. Developing, debugging, and running a Turbo C program is a snap.

Turbo C: The C compiler everybody's been waiting for. Everybody but the competition

Borland's "Quality, Speed, Power and Price" commitment isn't idle corporate chatter. The $99.95 price tag on Turbo C isn't a "typo," it's real. So if you'd like to learn C in a hurry, pick up the phone. If you're already using C, switch to Turbo C and see the difference for yourself.

System requirements

IBM PC, XT, AT or true compatibles. PC-DOS (MS-DOS) 2.0 or later. One floppy drive. 320K.

**Introductory price—good through July 1, 1987*

Technical Specifications

- ☑ *Compiler: One-pass compiler generating linkable object modules and inline assembler. Included is Borland's high performance "Turbo Linker." The object module is compatible with the PC-DOS linker. Supports tiny, small, compact, medium, large, and huge memory model libraries. Can mix models with near and far pointers. Includes floating point emulator (utilizes 8087/80287 if installed).*
- ☑ *Interactive Editor: The system includes a powerful, interactive full-screen text editor. If the compiler detects an error, the editor automatically positions the cursor appropriately in the source code.*
- ☑ *Development Environment: A powerful "Make" is included so that managing Turbo C program development is highly efficient. Also includes pull-down menus and windows.*
- ☑ *Links with relocatable object modules created using Borland's Turbo Prolog into a single program.*
- ☑ *ANSI C compatible.*
- ☑ *Start-up routine source code included.*
- ☑ *Both command line and integrated environment versions included.*

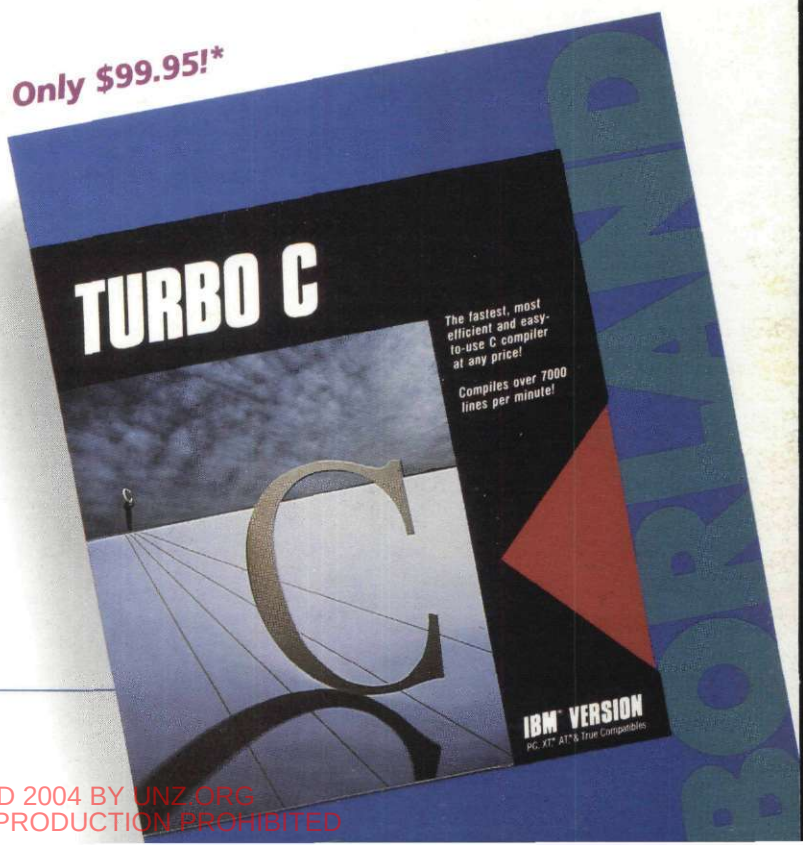
Turbo C and Turbo Pascal are registered trademarks and Turbo Prolog and Eureka: The Solver are trademarks of Borland International, Inc. Microsoft C and MS-DOS are registered trademarks of Microsoft Corp. IBM, XT, and AT are registered trademarks of International Business Machines Corp. *Copyright 1987 Borland International BI-1104*

Sieve benchmark (25 iterations)

	Turbo C	Microsoft® C	Lattice C
Compile time	**3.89**	16.37	13.90
Compile and link time	**9.94**	29.06	27.79
Execution time	**5.77**	9.51	13.79
Object code size	**274**	297	301
Price	**$99.95**	$450.00	$500.00

Benchmark run on a 6 Mhz IBM AT using Turbo C version 1.0 and the Turbo Linker version 1.0; Microsoft C version 4.0 and the MS overlay linker version 3.51; Lattice C version 3.1 and the MS object linker version 3.05.

Only $99.95!*


TURBO C
The fastest, most efficient and easy-to-use C compiler at any price!
Compiles over 7000 lines per minute!
IBM VERSION
PC, XT, AT & True Compatibles
BORLAND

BORLAND INTERNATIONAL
Vive la différence

4585 SCOTTS VALLEY DRIVE
SCOTTS VALLEY, CA 95066
(408) 438-8400 TELEX: 172373
TC 36

For the dealer nearest you or to order by phone call

(800)255-8008

in CA (800) 742-1133 in Canada (800) 237-1136

Number 3 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED